Detect and Identify Blocker Tags in Tree-based RFID Systems

Fei Wang*†, Bin Xiao†, Kai Bu†, and Jinshu Su*
*School of Computer, National University of Defense Technology, Changsha, China
Email: csfwang@comp.polyu.edu.hk, sjs@nudt.edu.cn.
†Department of Computing, The Hong Kong Polytechnic University, HK
Email: {csbxiao, cskbu}@comp.polyu.edu.hk.

***Abstract*—Blocker tags are initially introduced to protect regular tags in certain ID ranges, called *blocking ranges*, from unwanted scanning in RFID systems. But if misused, blocker tags can cause blocking attacks that corrupt the communication between interfered regular tags and readers. Previous approaches can only detect blocking behavior. However, they cannot distinguish malicious blocking from legitimate blocking that can be perfectly allowed to protect customer's privacy. To solve the problem, we carry out the first attempt in the paper to detect real blocking attacks by identifying malicious blocking ranges from authorized ones in a system. We present two pioneer probe-based protocols that can accurately identify malicious blocking ranges in popular tree-based RFID systems, and get rid of their impact before performing RFID applications. We validate the efficacy of the two protocols through theoretical analysis and simulation experiments. The results show that our protocols can identify blocking ranges very fast even when the blocker tag percentage is very low, for example, dozens of blocker tags among tens of thousands of regular tags. Our protocols deliver also a faster blocker tag detection than previous detection methods; our best protocol reduces detection time by over 90% compared with the state-of-the-art detection method.**

I. Introduction

Since Radio Frequency Identification (RFID) poses a potential threat to consumer privacy [1]–[5], blocker tags (called blocker for short) are introduced to selectively protect regular tags from unwanted scanning [6]. They transmit signals simultaneously with protected tags, violate anticollision protocols [7]–[9], and thus protect malicious readers from collecting data from protected tags. However, blockers can cause blocking attacks to RFID systems [6]. In such attacks, a blocker "protects" some regular tags that are not supposed to be protected, interrupting normal data collection from these tags. On the other hand, the blocker make readers believe presences of actually absent tags, causing a tag counterfeiting alike impact. Blocking attacks also result in inaccurate information collection, as well as more time and energy consumption.

Previous researches against blocking attacks lie mainly in detecting blockers in RFID systems. In [6], blockers are detected whenever scanned tags exceed the amount of tags in a system. P-BTD [10] is a probabilistic blocker tag detection method which detects blockers by comparing probabilities of interrogations' results. Above methods only tell whether blockers exist but not which tags are under their shields. In many cases, a simple existence indication of blockers may not help beat blocking attacks. Imagine in a public area with deployed RFID sensors for environmental data collection, legitimate blockers are authorized to protect certain tags, such as those on people's clothes. These blockers may be intentionally configured to cover tags on sensors, interrupting collection of environmental data. Under this circumstance, although previous methods can detect existence of blockers, they fail to distinguish legitimate blocking from malicious blocking, and thus cannot take further action to filter out attacking effect. To solve this critical problem, we propose detecting blocking attacks by identifying tag ID ranges blocked, called *blocking ranges*. Once blocking ranges are determined, we can tell which ranges are authorized, while others are not.

It is challenging to identify blocking ranges due to the huge ID space defined by the EPC standard [10]. It is also hard to determine whether a collision arises from a blocker or regular tags. Facing these challenges, this paper studies blocker detection and identification in tree-based RFID systems, for which blockers are initially introduced. To the best of our knowledge, this paper is the first to tackle not only blocker detection but also blocker identification.

In this paper, two novel protocols are proposed, Blocking Detection Polling protocol (BDP) and Fast Blocking Detection protocol (FBD) for different scenarios. BDP applies to scenarios with known tag IDs. By polling tag IDs, BDP detects affected tags and probes prefixes to identify blocking ranges. Its time complexity is linear with respect to the number of tags. FBD, on the other hand, adapts to scenarios with a large number of tags and without tag IDs as a priori. FBD detects a set of tag IDs together by probing their common prefix instead of per-ID probing like BDP. Leveraging tag population limitation, probes provide deterministic results about whether tags with a common prefix are affected by blockers. In FBD, a group responding mechanism is used to reduce the number of probes as well as detection time, which also minimizes search space of prefixes during identification. Analytical and experimental results show that both BDP and FBD can detect and identify all blockers without false positives. FBD achieves a significant improvement in time efficiency for both detection and identification. When compared with the state-of-the-art that only detects blockers [10], FBD reduces detection time by over 90%. The desirable efficiency of FBD makes it practically applicable in large RFID systems.

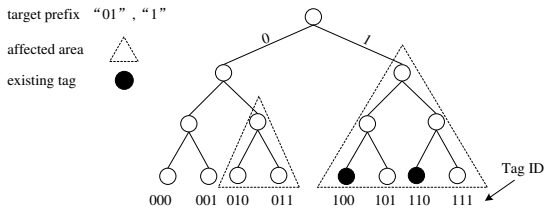

Fig. 1: Target prefixes and blocking ranges of blockers

II. Problem Statement

Blocker is first designed for tree-based RFID protocol [6]. When a reader broadcasts a prefix p, regular tags whose IDs begin with p emit the next bit of p, while blocker always emits both "0" and "1" bits with two antennas. Blocker itself causes a collision. Privacy invaders thus cannot infer valuable information from protected tags.

Normally, blocker is used to confront illegal readers. Configured with a particular tag ID range, called *blocking range*, a blocker activates when tags in the range are interrogated. However, blocker can illegally interfere with normal RFID applications, launching blocking attacks. We say a blocker is illegal if its actual blocking range is larger than what authorized. Illegal blocker affects RFID applications in two ways: obstructing readers from collecting data from present tags in its blocking range; imitating absent ones and making readers believe their existences. If no blocker is allowed in a system, present blockers are all illegal. But, considering situations of legal blockers being allowed, the only way to tell blocker's legitimate is identifying their blocking range (legal blockers may be misconfigured with illegal larger blocking range). We thus focus on blocking detection and identification, so as to deal with illegal blockers before RFID applications.

In this paper, we assume a blocker affects a subtree in T_{ID}. Namely, tag IDs in its blocking range are exactly leaves of a subtree. Their common prefix, called *target prefix*, corresponds to the root of affected subtree. For blockers not satisfying such assumption, we divide their affected areas into multiple subtrees. Each of them is caused by a blocker satisfying above assumption. For instance, we regard two blockers in Fig. 1. The reader thus perceives tags bearing IDs 010-111 are all present. In fact, only tags bearing IDs 100 and 110 exist. We use target prefix to identify blockers.

III. Blocking Detection Polling Protocol

We first present a Blocking Detection Polling protocol (BDP) to detect and identify blockers in RFID systems with known existing tag IDs, denoted by R_{ID}. BDP can be easily implemented using basic queries in tree-based protocol. BDP consists of a *polling phase* and a *determining phase* to find affected tag IDs by blockers and determine target prefixes respectively. In polling phase, the reader polls tags in R_{ID}. If a collision occurs, polled tag ID is affected by a blocker. We put all affected tag IDs into a set A.

In determining phase, we check prefixes of tag IDs in A to see which prefixes are affected by blockers. A prefix p_i is regarded as a target prefix if it is affected by a blocker and its one bit shorter prefix p_{i-1} is not. For an ID in A, we probe its prefixes sequentially from $(k-1)$-length prefix to shortest one, where k is the length of tag ID. Such process stops once a target prefix is found. Next, we explain how to decide a prefix is affected. Start with a prefix p_i of an tag ID id in A. If $N(p_i)$ represents a node in T_{ID} that corresponds to p_i, then $N(id)$ is a leaf node of T_{ID}. Obviously, $N(id)$ lies in one of $N(p_i)$'s subtrees. We know there is no tag (Case a) or at least one tag id' (Case b) in opposite subtree. In Case a, let the reader broadcast p_i. If a collision occurs, p_i is affected, as all tags whose IDs start with p_i emit the same bit. In Case b, check tags in the opposite subtree. If not all of them are in A, p_i is apparently not affected. Contrarily, we regard p_i as a suspicious prefix and probe prefixes of id'. If p_i is also regarded as suspicious, p_i is firmly affected. Otherwise, p_i is not affected if probes for id' ends before p_i.

BDP can accurately detect and identify blockers that affect existing tags. If no blocker, only polling phase is performed. Minimum execution time is $N \times (t_{tag} + t_s)$, where N is the number of existing tags, t_{tag} the time to transmit a tag ID and t_s the time of a tag's response. BDP can be finished in a short time when N is small. But, when N is large, it is not time efficient as vast prefixes need to be probed.

IV. Fast Blocking Detection Protocol

One way to accelerate blocker detection and identification is reducing reader's probes. In fact, a blocker always affects tags with same ID prefix (target prefix), such as tags attached to products of the same category. This gives us a fair chance to detect a set of relative tag IDs together. Given this, we present a Fast Blocking Detection protocol (FBD) that is more suitable for large RFID systems and totally ID-free. FBD significantly reduces readers' probes as well as search space of prefixes, and can thus detect and identify blockers fairly fast.

FBD consists of a *detecting phase* and an *identifying phase*. Once blockers are detected in detecting phase, the identifying phase is triggered to determine target prefixes. For acceleration, FBD follows two principles, detecting common prefixes instead of individual IDs, and probing a group of prefixes instead of one prefix per probe. A *Group responding protocol* (GRP) is used to achieve above principles in both phases.

Before describing FBD in detail, we would like to introduce a *tag population limitation mechanism*, which helps to obtain deterministic results from tags' responses, in regard to whether a prefix is affected by blockers. Specifically, when a reader broadcasts a prefix p, only tags whose IDs begin with $p0$ (or $p1$) respond. They all emit the same bit "0" (or "1"). We thus tell p is affected if a collision occurs. Filter capability and select command in EPC specifications [11] can be used to achieve such flexible limitation. In FBD, a particular limitation strategy ("0" or "1"), is announced each time when the reader broadcasts a query. We present FBD against general blockers that always emit "0" and "1" simultaneously.

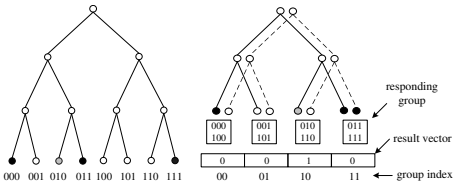

Fig. 2: A group responding round with $l = 3$ and $c = 2$. A leaf node represents the root node of a subtree. Black leaf node indicates that there exists at least one tag in the subtree. The blocker's target prefix is "010".

A. Group Responding Protocol

Given huge prefix space, one prefix per probe is exhausting. *Group Responding Protocol* (GRP) solve the problem by dividing prefixes into *responding groups* and probing a group of prefixes by broadcasting group index.

Each *Group Responding Round* (GRR) focuses on prefixes of the same length, denoted by P_l where l is the length. By specifying continuous c bit positions in prefixes, we divide P_l into 2^c groups. Prefixes with the same bits at the same bit positions belong to the same group. The c bits are combined orderly as group index. A tag only participates in an interrogation when its prefix belongs to interrogated group. We call l *query level* and c *group parameter*.

There are two ways to perform GRRs. One is to probe all groups one by one with a *single group-probing query* (SIGQ). The query includes l, c and group index g. It actually probes all prefixes in g. Given c bit positions, a tag decides whether to emit the $(l+1)^{th}$ bit of its ID. Under population limitation, tags that respond emit the same bit. Therefore, if a collision occurs, blockers do exist and at least one prefix in g is affected. A more efficient way to perform GRRs is to let tags automatically respond in continuous time slots, without reader's intervention. The reader broadcasts a *slotted group-probing query* (SLGQ) without specific g. Following 2^c time slots are assigned to 2^c groups. Tags emit the $(l+1)^{th}$ bit of their IDs in slots of their own groups, if they should respond. Other tags sleep for 2^c slots and then wake up to receive next query. The reader monitors time slots, collided or not, and figures out groups including affected prefixes. Fig. 2 illustrates a GRR upon P_3. Three of eight prefixes in P_3 are present in existing tag IDs. Using the second and third bits as group index, a collision occurs in the third slot after a SLGQ. That means one or more prefixes in group "10" are affected by blockers. Using a series of SIGQs can arrive at exactly the same conclusion.

SIGQ is more adaptable to situations when only a few groups is of interest. For instance, if a few categories of products exist in a warehouse, we can use several SIGQs to probe groups including existing prefixes. When groups of interest are excessive, a SLGQ can reduce detection time as well as bits that regular tags receive. In FBD, we use SLGQs in detecting phase and SIGQs in identifying phase, as fewer prefixes are involved in identifying phase.

B. Detecting Phase

In detecting phase, the reader performs multiple GRRs to detect collisions caused by blockers. Each GRR focuses on a distinct query level. The challenge is how to choose query levels, so as to balance complexities in blocker detection and identification. Performing one GRR on the lowest level k is a quick way to detect blockers. But, it increases time cost in identifying phase. On the other hand, performing GRRs on every level radically simplifies identification, and yet it unnecessarily increases detection time. In fact, a level-l GRR can expose all blockers whose target prefixes are not longer than l. For example, the blocker in Fig. 3 has a target prefix less than l and its impact (the triangle with diagonal lines) is observed in level-l GRR. We thus wisely choose query levels of GRRs using a variant of binary-based algorithm. Let $[l_{high}, l_{low}]$ represent the undetermined part of T_{ID}, initially being $[0, k]$. Then the next query level is

$$l_{i+1} = \begin{cases} (l_{high} + l_{low})/2 & |l_{low} - l_{high}| > c \\ l_{low} & else \end{cases} \quad (1)$$

l_{high} and l_{low} depend on result of i^{th} GRR. If no collision, $l_{high} = l_i$. If all slots are collided, we set $l_{low} = l_i$ to perform a higher-level GRR. Otherwise, the identifying phase starts.

C. Identifying Phase

In identifying phase, we gradually narrow search space of prefixes until finding target prefixes. Based on the result of trigger GRR (on level l), we selectively choose prefixes to probe, accelerating the identifying process.

We use a vector V to represent result of level-l GRR. A bit "1" in it indicates a collided slot. We define *segment* as a nonzero subvector of V, in which all corresponding collided slots are caused by one blocker. It is called *z-segment* if its length is 2^z. For example, there are a 2-segment "1111" and a 0-segment "1" in "V=11111000". For each segment, we selectively probe prefixes as follows to infer target prefixes.

1) Rough Location: Due to group responding, a segment corresponds to 2^{l-c} different areas in T_{ID} (grey triangles in Fig. 3). Let $A^{(s_i)}$ be one of these areas, where s_i represents the path from the root of T_{ID} to the highest node in $A^{(s_i)}$. We roughly locate areas where blocking occurs by broadcasting a SIGQ with s_i as the group index and l the query level. If the reader detects a collision, then $A^{(s_i)}$ is the area we are looking for. At least one prefix in $A^{(s_i)}$ is affected by blocker. We may find several such areas for one segment. That is because effects of more than one blocker are overlapped on the same groups.

2) Accurate Determination: Suppose $A^{(s_i)}$ is the area obtained in rough location. There are 2^z possible target prefixes in it. Their lengths range from $l - z$ to l. We let the reader probe these prefixes orderly from top to down, as it does in standard tree-based protocol, only population limitation is in use. If a collision is detected, we find a target prefix and ignore other longer prefixes. When all prefixes in $A^{(s_i)}$ are detected, we can identify all blockers affecting this area. In Fig. 3, the reader finds a target prefix s_m1 in $A^{(s_m)}$ using as few as five basic queries. Most of time, the target prefix can be found with one probe, as the probability that more than one target prefixes coexist in an area is very low.

Identifying blockers is a process consumptive of time and energy. In some cases, we only detect prefixes we are inter-

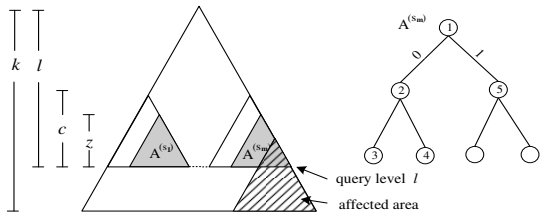

Fig. 3: The identifying phase. Nodes represent prefixes that the reader probes. Numbers represent the probing order. $m = 2^{l-c}$ and $z = 2$.

ested in, thus significantly reducing number of probes in both steps. For instance, given existing tag IDs or categories, we do not need to dig into an area without existing prefixes for target prefix searching.

D. Complete Detection and Identification

Next, we discuss how to detect and identify all blockers in a system. The situation occurs when multiple blockers actually exist or a blocker does not satisfy subtree assumption. After identifying phase, blockers with greater blocking ranges can be identified. To detect other blockers, more GRRs on lower levels are needed. Question is, identified blockers may affect following detection due to affected area overlapping. We address this problem by dividing T_{ID} into multiple subtrees and performing FBD on each subtree. Once a blocker is identified, we prune its affected area (a subtree) from T_{ID} and divide the rest part into multiple pending subtrees, whole or parts of which have not been detected. Then we recursively perform FBD on these subtrees. Algorithm 1 describes the whole process of complete detection and identification.

Algorithm 1 Complete detection and identification

1: Set $T^{(s)} \leftarrow T_{ID}$, $l_{high} \leftarrow 0$, $l_{low} \leftarrow k$, $c = c_0 - |s|$, $Set_A \leftarrow \emptyset$.
2: Perform FBD on $T^{(s)}$. Suppose identifying phase is triggered by a level-l GRR and results in a set tmp_A of target prefixes.
3: Extract pending subtrees from $T^{(s)}$ based on tmp_A, and add tmp_A into Set_A.
4: For each pending subtree $T^{(s')}$, set $T^{(s)} \leftarrow T^{(s')}$, $l_{high} \leftarrow l$, $l_{low} \leftarrow k$ and execute step 2-4.
5: **return** Set_A

E. Analysis of FBD

In this section, we analyse FBD theoretically and show how to set c_0 to minimize overall execution time.

Suppose that θ blockers are identified and lengths of their target prefixes in descending order are $\rho_1, \rho_2, ...\rho_\theta$. Then we need N_s time slots to perform level-l GRRs for all extracted pending subtree.

$$N_s \approx 2^{c_0} \times (1 - \frac{1}{2^{\rho_1}} - \frac{1}{2^{\rho_2}} - \cdots - \frac{1}{2^{\rho_\theta}}) \quad (2)$$

Let $a = m/2^k$, representing proportion of affected tag IDs by blockers. By (2), we have $2^{c_0}(1-a) \leq N_s \leq 2^{c_0}$. When a is very small, $N_s \approx 2^{c_0}$. Our numerical result proves a has no effect on optimal c_0 when $a < 0.4$. To simplify analysis, we assume only one blocker is exposed in last GRR and it causes a z-segment of collisions. The segment corresponds to 2^{l-c_0} subtrees in T_{ID}. The probability that tags exist in such a subtree is $1 - ((2^l - 2^z)/2^l)^N$. The expected number of subtrees that the reader needs to probe is

$$U \approx 2^{l-c_0} \times (1 - e^{-\frac{N}{2^{l-z}}}). \quad (3)$$

Let t_{sl}, t_{si} and t_b be times for a reader to transmit a SLGQ, SIGQ and basic query, respectively. Then execution time is

$$T = \alpha t_{sl} + N_s t_s + U \times (t_{si} + t_s) + \beta(t_b + t_s), \quad (4)$$

where α and β are factors. The first and last terms are the time to transmit SLGQs and the execution time of accurate determination. Compared with the other two terms, they are both negligible. Regarding T as a functions of c_0, we can find optimal c_0 that minimizes T from $\frac{dT}{dc_0} = 0$. We calculate T with different l, c_0, and z. Interestingly, T shows insensitivity to l but is easily influenced by z. We show numerical value of T with respect to c_0 in Fig. 4. When performing FBD on a subtree $T^{(s_i)}$, z is no more than $c_0 - |s_i|$. Especially, when $|s_i| > c_0$, c is set to 1. Accordingly, if some blockers are exposed in a GRR, we have $z = c = 1$. When a is small, the situation occurs with high probability. We thus prefer the optimal c_0 when $z = 1$. When N is 5,000 and 50,000, the optimal c_0 are 8 and 10.

V. Evaluation

We evaluate performance of BDP and FBD. Actual parameters in practical RFID systems [12] are used in simulations.

A. Detection Time

Since threshold-based method [6] and P-BTD algorithm [10] only do detection job without using existing tag IDs, we only compare detection time of FBD with them. To be fair, we use 16-bit tag ID to evaluate FBD under the same conditions with [10]. As shown in Fig. 5, the reader performs 2.6 GRRs on average before it detects existences of blockers. Compared with P-BTD, the better existing method, FBD reduces detection time by more than 90%. We also consider energy costs of different methods (Fig. 6). Received bits per tag in FBD is several orders of magnitude fewer than P-BTD (tag responses are almost equal). In summary, FBD reduces both time and energy costs in detection. We compare FBD with BDP in Fig. 5. Detection time of FBD does not change much as a increases, greatly outperforming BDP.

B. Overall Execution Time

Next, we show how parameters affect the overall execution time of blocker detection and identification.

1) Vary the number N of regular tags: We show simulation results in Fig. 7. When a becomes smaller, overall execution time of BDP gets closer to its minimum execution time while that of FBD is much less than it all along. Compared with BDP, FBD reduces execution time by more than a half and the degree reaches up to one magnitude as N increases. Due to insensitivity to N, FBD meets practical and scalable requirements of large RFID systems with thousands of tags.

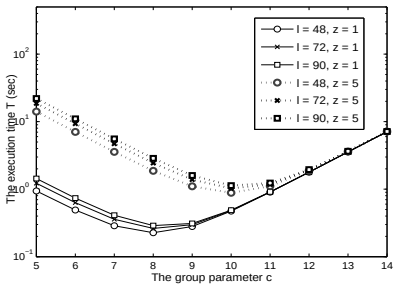

Fig. 4: Execution time for a particular query level with respect to the group parameter c

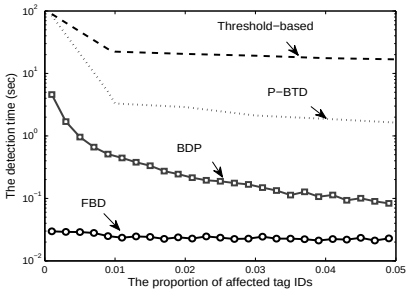

Fig. 5: The detection time with respect to a

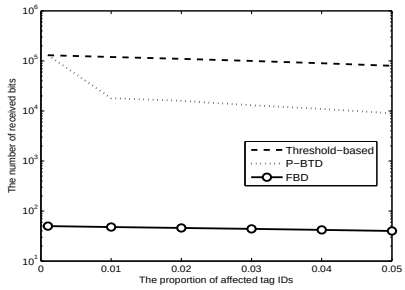

Fig. 6: The number of received bits with respect to a

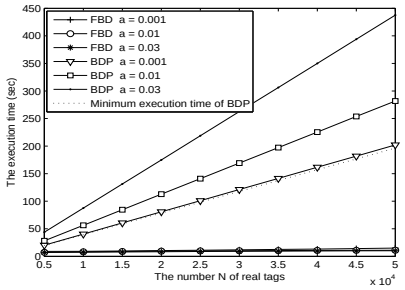

Fig. 7: The execution time with respect to N.

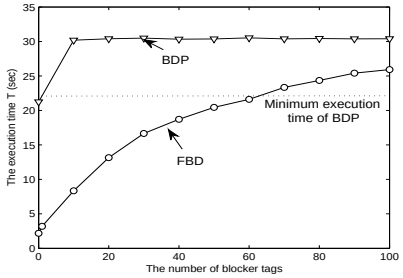

Fig. 8: The execution time with respect to the number of blockers

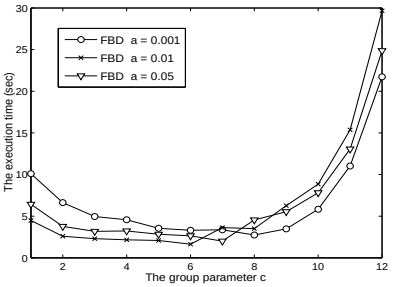

Fig. 9: The execution time with respect to c_0

2) Vary the number of blockers: We analyze the influence of blocker amount on execution times of FBD and BDP. Given a fixed a (0.001, 0.01, 0.03 and 0.05), we change the number of blockers from 0 to 100. The simulation result indicates that overall execution time of FBD increases with the number of blockers. But, with a reasonable blocker amount (e.g. less than 60), execution time is still less than minimum time of BDP. In Fig. 8, we show simulation result of one case ($N = 5,000$, $a = 0.01$). Similar results are found in other cases.

3) Vary group parameter c_0: Fig. 9 shows the execution time of FBD with respect to c_0. When $c_0 > 10$, execution time increases sharply. That is because the reader has to wait for 2^{c_0} time slots in each GRR, even though a great portion of GRRs do not expose collisions. When c_0 is very small, the execution time is also a little higher, as it takes much time to find affected prefixes in a large responding group. The minimum value of execution time appears when c_0 is between 6 and 8 ($N = 5,000$). We notice a slight difference between simulation results and numerical analysis. The reason is that some GRRs are not followed with an identifying phase. In these GRRs, a smaller value of c_0 saves much time. Since we cannot predict results of GRRs, a trade-off value of c_0 should be used. The larger c_0 means fewer responses per tag. We thus prefer larger c_0 to balance time and energy cost.

VI. Conclusion

In this paper, we study the problem of detecting and identifying blockers in tree-based RFID systems. Two protocols, BDP and FBD, are proposed for different scenarios. FBD is more efficient in large RFID systems. It takes much less time than previous methods to detect blockers. Our protocols identify accurate blocking ranges of blockers in a short time, which has not been done before. We thus can distinguish legitimate blockers from illegal ones, and can totally remove their malicious impact. Our protocols will be of great use in tree-based RFID systems to defeat blocking attacks.

Acknowledgment

This work was partially supported by the National High Technology Research and Development Program of China (863 Program) No. 2011AA01A103, NSFC under Grant No. 61202488, and the project HK RGC PolyU 5286/12E.

References

[1] F. Niederman, R. G. Mathieu, R. Morley, and I.-W. Kwon, "Examining RFID applications in supply chain management," *Commun. ACM*, vol. 50, pp. 92–101, July 2007.
[2] K. Bu, B. Xiao, Q. Xiao, and S. Chen, "Efficient pinpointing of misplaced tags in large RFID systems," in *SECON*, 2011, pp. 287–295.
[3] M. Kodialam, T. Nandagopal, and W. C. Lau, "Anonymous tracking using RFID tags," in *INFOCOM*, 2007, pp. 1217 –1225.
[4] Q. Xiao, K. Bu, and B. Xiao, "Efficient Monitoring of Dynamic Tag Populations in RFID Systems," in *EUC*, 2011.
[5] S. Chen, M. Zhang, and B. Xiao, "Efficient information collection protocols for sensor-augmented RFID networks," in *INFOCOM*, 2011, pp. 3101–3109.
[6] A. Juels, R. L. Rivest, and M. Szydlo, "The blocker tag: selective blocking of RFID tags for consumer privacy," in *CCS'03*, pp. 103–111.
[7] D. Hush and C. Wood, "Analysis of tree algorithms for RFID arbitration," in *Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on*, 1998, p. 107.
[8] B. Zhen, M. Kobayashi, and M. Shimizu, "Framed ALOHA for multiple RFID objects identification." *IEICE Transactions*, pp. 991–999, 2005.
[9] S.-R. Lee, S.-D. Joo, and C.-W. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," in *MobiQuitous*, 2005, pp. 166–172.
[10] E. Vahedi, V. Shah-Mansouri, V. Wong, I. Blake, and R. Ward, "Probabilistic analysis of blocking attack in RFID systems," *Information Forensics and Security, IEEE Transactions on*, pp. 803 –817, 2011.
[11] D. M. Dobkin, *The RF in RFID: Passive UHF RFID in Practice*. Elsevier, 2008, ch. 8.
[12] P. Semiconductor, "I-code smart label RFID tags," *http://www.nxp.com/acrobatdownload/other/identification/SL092030.pdf*, January 2004.