# FastProbe: Malicious User Detection in Cognitive Radio Networks Through Active Transmissions

Tarun Bansal, Bo Chen and Prasun Sinha
Department of Computer Science and Engineering, Ohio State University, Columbus, Ohio 43210
Email: {bansal, chebo, prasun}@cse.ohio-state.edu

***Abstract*—Sensing white space channels to detect whether a particular channel is free or not is very crucial to the operation of Cognitive Radio Networks (CRNs). Cooperative sensing has been shown to improve the performance of channel sensing. However, cooperative sensing is susceptible to malicious users that may not faithfully follow sensing instructions to save energy and/or time, or to launch denial of service attacks against the network. In this paper, we propose a novel active transmissions based algorithm, *FastProbe* for detecting malicious users. FastProbe proactively detects malicious users before the CRN causes any interference to the Primary Users. Further, using active transmissions, FastProbe achieves higher detection accuracy while maintaining lower overheads when compared with existing algorithms. Simulations and experiments show that in the presence of malicious nodes operating under 2 different attack models, FastProbe reduces the throughput loss due to sensing by as much as 65% compared to existing algorithms.**

## I. Introduction

FCC regulations require Cognitive Radios (CRs) to operate on only those white space channels that are not occupied by Primary Users (PUs). To that end, CRs need to perform regular *in-band* scanning (or sensing) [1] of channels to avoid interference to PUs as well as *out of band* sensing to find unused channels. To achieve better sensing accuracy, CRs perform cooperative sensing where multiple nodes participate in channel sensing and their sensing results are processed centrally at a *sensing server* to determine if the channel is occupied [2]. Cooperative sensing improves sensing accuracy but at the same time makes the system more susceptible to malicious users that may be present in the system. A malicious user may (i) have a hardware error due to which its readings are erratic; (ii) be hacked by the owner so that it reports arbitrary sensing results without performing any sensing to save time and/or energy; (iii) report the channel to be busy so as to either leave the channel free for its own personal usage [3] or to initiate a Denial-of-Service attack against the network [4]; (iv) selfishly report the channel to be free so that it can operate on that channel; or, (v) skip in-band sensing in order to save energy and/or throughput. The recent increase in attacks against TCP Protocol [5], mobile devices and other hardware devices [6] as well as the software based design of CRs suggests that in the future CRs will be very vulnerable to similar kinds of attacks. Such attacks may affect the sensing capability of CRs and may result in nodes causing interference to PUs, thereby violating the FCC requirements.

Sahai et *al.* [2] have shown that cooperation among 10-15 *non-malicious* nodes is required to achieve good sensing results. However, the presence of even a few malicious users in the system causes this number to grow significantly [2], thereby severely affecting the performance of cooperative sensing systems. Hence, it is beneficial to detect malicious users and then remove their readings when aggregating the sensing results. The problem of detecting such malicious users has been referred to in the literature as Secure Sensing Data Falsification (SSDF) problem [7].

CRs participate in two types of sensing: (i) Out of band sensing: This is primarily used for finding out which channels are not occupied by PUs. Such channels are used for communication if the existing channels in use are occupied by PUs. (ii) In-band sensing: This is used to detect the arrival of the PU on a channel that is currently being used for communication by CRs. FCC regulations require CR to leave the channel within 2 seconds of arrival of the PU. Thus, the problem of SSDF also has two variants: (i) Detecting malicious users that report incorrect results for out of band sensing; and, (ii) Detecting malicious users that do not faithfully perform the in-band sensing.

One of the most common techniques used for detecting malicious users is based on the assumption that neighboring CRs have similar readings. These algorithms utilize a *passive approach* that detects malicious users at the same time while sensing the channel for the presence of PUs. The algorithms based on this technique collect readings from all CRs and then mark those nodes as malicious whose readings differ significantly from their neighbors. Existing techniques based on this approach have three major drawbacks:

- Firstly, since the actual state of the PU is unknown to the system, therefore, the absence of this ground truth makes it much harder for existing algorithms to detect malicious users. Thus, if multiple malicious CRs are present in the system, it is possible that before they are detected, they are able to convince the *sensing server* to make incorrect decisions about the occupancy of the channel. This may lead to CRs either causing interference to the PU transmissions or incorrectly deducing a channel to be occupied resulting in lower throughput.
- Secondly, in practice due to the presence of obstacles and multipath [2], [8], the assumptions that neighbors have similar readings or the readings follow a particular correlation model do not hold well [9] resulting in either high false positives or high false negatives even if precise location of CRs is known. For example, in Figure 1, nodes in Group 1 can only sense $PU_1$ while nodes in group 2 can only sense $PU_2$ though nodes

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

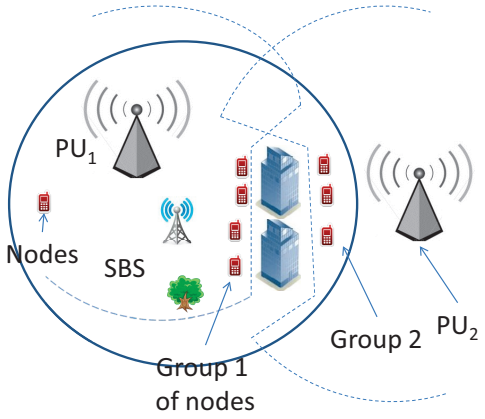


Fig. 1. Nodes in Group 1 can only sense $PU_1$ while nodes in group 2 can only sense $PU_2$. Dotted lines show the range of $PU_1$ and $PU_2$ within which they can be detected by CRs.

in both groups are neighbors of each other. Since the readings of nodes in group 2 do not match with the majority of their neighbors, therefore these algorithms tend to mark nodes in Group 2 as malicious. This may prevent the CRN from sensing $PU_2$, thereby causing interference to its transmissions.

- Finally, the current algorithms do not allow detection of malicious users that do not perform *in-band sensing*. This is because as soon as the PU arrives on a channel, it becomes difficult for the existing algorithms to differentiate between the transmissions of malicious users and the actual primary users.

In this paper, we propose *FastProbe*, a novel *active transmissions* based algorithm for solving SSDF problem that utilizes Primary User Emulation (PUE) signals. In FastProbe, CRs are subjected to various *sensing tests* by the sensing server whereby a neighbor of node to be tested transmits PUE signals. On the basis of the received signal strength report obtained from the node being tested, it is possible to estimate whether this node is malicious or not. *The tests can be actively delivered to nodes before the actual sensing needs to be done, allowing FastProbe to proactively detect malicious users.* Also, our detection process is more accurate because of the following four reasons: (i) Sensing Server has complete knowledge about the ground truth (*e.g.* transmission power level) for the tests, thus it can more accurately conclude if the received power level reported by a receiver is correct or not; (ii) Readings of received signal strength at a receiver for a given transmitter are compared with the previous readings for the same transmitter-receiver pair, thus removing the uncertainties that may arise if the signal loss models are not followed; and, (iii) It does not incorrectly mark those nodes as malicious that are located behind obstacles. FastProbe can also detect those malicious users that do not perform in-band sensing. Finally, because of higher accuracy, FastProbe has lower overhead since malicious nodes are detected faster in fewer iterations.

The problem of detecting malicious nodes in a CRN raises multiple challenges:

- Signal loss models (*e.g.*, path loss) may not hold [9] due to obstacles, multipath etc. Further, a node that is located behind an obstacle is hard to distinguish from a malicious node leading to inaccurate detection of malicious nodes.
- A malicious node that does not perform in-band sensing is hard to detect since it may emulate PU signals making it difficult to distinguish it from the actual PU.
- Detection of malicious nodes generally occurs at the cost of throughput. To minimize the loss in throughput, it is beneficial to detect malicious nodes efficiently.

This paper is organized as follows. Section II describes the relevant related work. Section III describes the system model used. The next section explains *FastProbe* in detail. Section V discusses how FastProbe can be made more robust. Simulation and experiment results are presented in Section VI. Finally, we conclude this paper in Section VII.

## II. Related Work

The problem of detecting malicious users has been extensively studied before. An algorithm proposed by Min et *al.* [7] first estimates the transmission level of the PU and the path-loss exponent on the basis of sensing readings of all CRs. However, their algorithm requires knowledge of the exact locations of CRs. Further, they assume that the path loss exponent is same for all CRs which may not be true depending on exact location of obstacles [8], [9]. Another algorithm, ADSP, proposed by Min et *al.* [10], [11] for secure sensing works by arranging CRs in clusters. It is assumed that nodes in the same cluster have high correlation among themselves, and here also, knowledge of distance between all pair of nodes is required. Kaligineedi et *al.* [3] have used outliers detection technique that detects malicious users by comparing the report of users with that of its neighbors. Similarly, algorithms such as [3], [7], [10]–[13] that are based on the assumption that neighboring nodes have similar readings suffer from the following shortcomings:

- Since these algorithms make use of redundant readings from neighboring nodes to detect outliers, therefore the neighbors are also required to sense the channel. However, this results in higher communication overhead and lower throughput as the neighboring CRs are required to stop communication to perform sensing.
- Presence of obstacles in the field may affect correlation of readings among neighboring nodes, resulting in incorrect labeling of malicious nodes.
- Since the reading of a node is compared with that of its neighbors, therefore, for accurate detection of malicious users, it is desired that all CRs have high degree.
- Using current algorithms to detect malicious nodes that do not perform in-band sensing is not possible if the PU is inactive on that channel.

Algorithms proposed by Wang et *al.* [14] work for only prespecified attack models. Further, they assume that spectrum occupancy is same at all nodes. This assumption may not

be always true since Cognitive Radio Networks (CRNs) can have a range of as much as 100 km [4]. Similarly, algorithms proposed in [13], [15], [16] assume the spectrum occupancy to be same for all nodes. Another algorithm proposed by Fatemieh et *al.* [4] requires collecting data from every node in a war-driving like fashion during the network setup which is generally a tedious process and may need to be repeated periodically.

## III. System Model and Objectives

We assume an infrastructure based network of CRs where multiple nodes (or Secondary Users, SUs) may be associated with a SU Base Station (SBS) and that the location of SUs is unknown. For the sake of exposition, we assume the existence of only one SBS, however, FastProbe can be easily extended to multiple SBSs as discussed in the Technical Report [17]. We assume that SBS facilitates cooperative sensing by sending control messages that may be piggybacked on other control messages or data messages. All the control messages from SBS to SUs and from SUs to SBS are assumed to be encrypted that can only be decrypted by the intended recipient. SBS is assumed to be not compromised since it is generally installed by an expert. We assume that it is possible for an SU to emulate PU transmissions [18], [19] and malicious users cannot distinguish if a neighboring SU or a real PU is transmitting (see Section V-4).

The power of a wireless signal transmitted from a node $n_i$ to another node $n_j$ degrades due to multiple factors such as propagation loss, absorption loss (due to slow fading), multipath loss and losses due to noise such as thermal noise. For a given transmitter and receiver, slow fading losses such as propagation loss, absorption loss and multipath loss remain constant. It has been shown [20], [21] that in cognitive radios, fast fading losses due to multipath can be neglected due to frequency diversity (computing path loss over multiple subcarriers and taking an average) as well as the time diversity (taking multiple readings over the same subcarrier and computing its average). Therefore, we assume that after compensating for fast fading losses, in time domain, the path loss between any two stationary Cognitive Radios (CRs) variation can be modeled using a Gaussian distribution [22] with zero mean and $\sigma^2$ variance.

When there is no transmitter on a particular channel in the whole network, then upon sensing that channel, the received power level that a SU detects is called its *noise floor level*. The noise floor level of a node may vary due to thermal noise etc. For the sake of exposition, we assume that distribution of noise floor level at all SUs has the same mean (denoted by $\psi$) and variance (denoted by $\sigma^2$) which are known to the SBS. However, it is also possible to learn these parameters during the node initialization period (see Section IV-C). An SU $n_i$ is said to be a neighbor of $n_j$ if the power level at $n_i$ is more than the upper $95\%$ confidence interval of noise floor level of $n_i$ (denoted by $\psi^+$) when $n_j$ is transmitting at its highest power level. We denote the set of neighbors of $n_i$ by $N(n_i)$. For a given SBS, $\mathcal{G}$ denotes the directed graph whose vertices are the SUs and whose edges represent the neighborhood relationship. Table I gives the list of commonly used symbols in this paper.

The malicious SU may deliberately send incorrect sensing results to SBS in order to save energy and/or to increase

TABLE I. Symbols Used

| Symbol | Meaning |
|---|---|
| $n_i$ | Node (or SU) $i$ |
| $c_k$ | Channel $k$ |
| $\rho_i$ | Reputation value of $n_i$ |
| $\overline{\rho}$ | Average $\rho$ over all nodes |
| $\psi$ | Noise floor of SUs (in dB) |
| $\psi^+$ | Upper 95% confidence interval of noise floor of SUs (in dB) |
| $\omega$ | Reputation value threshold below which all SUs are considered as malicious |
| $T_i$ | Observed transmission power level of $n_i$ |
| $T_i^e$ | Transmission power level of $n_i$ as expected by SBS |
| $R_j$ | Observed power level received at $n_j$ |
| $R_j^e$ | Received power level at $n_j$ as expected by SBS |
| $P_{ij}$ | Observed path loss from $n_i$ to $n_j$ |
| $P_{ij}^e$ | Path loss from $n_i$ to $n_j$ as expected by SBS |

throughout as discussed in Section I. The SBS maintains reputation value $\rho_i$ for every SU $n_i$. Once the reputation value of an SU drops below a certain threshold ($\omega$), the SBS may choose to penalize the SU by excluding it from cooperative sensing and/or reducing its throughput.

It has been shown that cooperative sensing (or scanning) increases sensing accuracy [2]. When the SBS wants to ascertain the occupancy of a channel, it sends a sensing request to a subset of its associated SUs. After sensing the channel, SUs report the measured power level back to the SBS that aggregates the received results to finally determine the channel occupancy. Due to its simple implementation and wide-spread acceptance [1], we assume that SUs use energy-detection methods for sensing.

*Objectives:* It is desired that an algorithm for detecting malicious nodes should have high detection accuracy, fast detection time, low overhead, low false positives and should lead to high PU sensing accuracy as well. Further, it should detect malicious nodes before they cause any interference to the PUs as well as those malicious SUs that do not perform in-band sensing. In the next section, we will describe our algorithm, *FastProbe* that satisfies all the required properties.

## IV. Algorithm FastProbe

In this section, we will describe our algorithm, FastProbe by first giving its overview, followed by description of the optimization problem that we are solving. Next, we will describe how node initialization is done in FastProbe. After that, we will give a detailed description of FastProbe.

### A. Overview

In FastProbe, the SBS detects malicious SUs by giving *sensing tests* to them. Using the reports from the sensing (or scanning) tests, SBS also maintains path loss between every pair of neighbors. When a node (say $n_j$) is to be tested, SBS asks one of its neighbors (say $n_i$) to transmit PU Emulation (PUE) signals at a specified power level and $n_j$ is asked to sense on the same channel. As PUE signals can't be distinguished from real PU transmissions (see Section V), therefore a malicious node would most likely report incorrect

sensing results for some of the sensing tests as well. Assuming $n_j$ reports incorrect sensing result when it is being tested by $n_i$, then the path loss computed from $n_i$ to $n_j$ using $n_j$'s report would be inconsistent with the path loss value maintained by SBS using the previous reports. However, if $n_j$ is not malicious, then the path loss from $n_i$ to $n_j$ computed from this report and all future reports will follow Gaussian distribution with the mean approximately same as the one maintained by the SBS and variance $\sigma^2$ (see Section III). Thus, on the basis of received signal strength reports from sensing tests, it is possible for the SBS to detect malicious SUs.

However, there is a possibility that a malicious SU is selected for transmitting PUE signals and it does not transmit at the specified power level in order to get other SUs labeled as malicious. In order to detect such malicious SUs, FastProbe first verifies if the testing SU transmitted at the specified power level (See Section IV-D for details).

### B. Optimization Problem

In order to minimize the loss in throughput, it is desired that SUs should perform sensing with little overhead. In FastProbe, at regular intervals, the SBS computes a set of SUs to be tested (say $A$) and a testing schedule, $S$. The schedule $S$ is a set of tuples $(n_i, n_j, T_j, t_l)$ representing that $n_j$ tests $n_i$ in slot $t_l$ by transmitting at a power level $T_j$. Schedule $S$ is said to be *error-free* if a node (say $n_i$) is being tested in some time slot in $S$, then exactly one neighbor of $n_i$ is transmitting in the same slot. In one time slot, multiple nodes can transmit simultaneously, however, testing all the nodes in the set $A$ may require multiple time slots. When the testing is ongoing on channel $c_k$, it is not possible for any node in the system to use $c_k$ for transmitting data and therefore, to minimize this loss in throughput, it is beneficial to minimize the length of $S$. The problem of computing the minimum length testing schedule can be written as:

$$\text{Problem} \quad : \quad \min_{S \in \mathbb{S}} \max(t_i : \exists n_a, n_b, T_b : (n_a, n_b, T_b, t_i) \in S) \tag{1}$$

subject to:

$$\forall n_a \in A, \exists n_b, T_b, t_i \quad : \quad (n_a, n_b, T_b, t_i) \in S \tag{2}$$

$$S \; is \; error - free \tag{3}$$

A naive method would be to test all nodes from SBS in a single time slot using high transmission power so as to reach all SUs. However, it has several disadvantages: (i) Some SUs may behave maliciously only when received signal strength from PU is low. However, high transmission power of SBS would result in reception power level being always high at SUs that are located close to the SBS, thus making it hard for the SBS to test such SUs; (ii) If the SBS participates in testing, then no SU can send data to the SBS in that slot, resulting in low throughput for the whole system; and, (iii) A transmission made by SBS could be easily distinguished by malicious users from actual PU transmissions because of its transmission power level being approximately same as that of its actual data transmissions. In the general case when SBS is not used for testing, the problem is NP-Hard as proven in the technical report [17].

### C. Node Initialization

In order to detect malicious SUs, the SBS needs to maintain path loss between all pairs of neighbors. However, since path loss is dependent on transmission frequency, therefore one naive solution that has high overhead is to maintain path loss for every pair of neighbors for all possible central frequencies. In this paper, we assume that the path loss between a pair of nodes varies approximately linearly with the log of frequency [9], [23], therefore using the following equation, and the values of $a_{ij}$ and $b_{ij}$, it is possible to arrive at expected path loss ($P^e_{ij}$) from $n_i$ to $n_j$ for any frequency $f$:

$$P^e_{ij} \approx a_{ij} \log f + b_{ij} \tag{4}$$

The value of $a_{ij}$ and $b_{ij}$ have been shown to depend on the environment [23], therefore FastProbe computes it as it proceeds. In order to use the above equation, SBS computes an *initial value* of $a_{ij}$ and $b_{ij}$ during the node initialization phase. Observe that if the receiver is malicious, it is possible that it may report incorrect signal strengths. However, in FastProbe, the receivers are not aware of the identity of the transmitters or the transmission power level chosen by them. This makes it easier for the SBS to detect such malicious nodes. This is further discussed in Section V-3.

At any time, let $Y$ be the set of nodes for whom neighbor information (*i.e.* set of neighboring nodes, $a_{ij}$ and $b_{ij}$ with all neighbors) is not available. When the network is initiated, $Y$ will include all the SUs, otherwise $Y$ will include the recently joined nodes. In order to obtain the neighbor information for a node, say $n_i$ in $Y$, the SBS asks $n_i$ to transmit at two different frequencies in two time slots using its highest transmission power while other SUs in the network silently listen. From the readings of the other SUs (say $n_j$) that were listening, SBS can compute the path loss from $n_i$ to $n_j$ for two different frequencies which can be further used to compute $a_{ij}$ and $b_{ij}$ in (4) using "Weighted Linear Least Square Method" with higher weight given to recent readings. However, computing values of $a_{ji}$ and $b_{ji}$ requires multiple slots since all other nodes need to transmit one-by-one while $n_i$ silently listens. In order to reduce the time taken, we initially set $a_{ji}$ same as $a_{ij}$ and $b_{ji}$ as $b_{ij}$. Studies [24] have shown that at high transmission power levels, around 90% of the links are symmetric even if neighboring nodes are transmitting on the same frequency simultaneously. Here in FastProbe, $n_i$ is the only node that transmits on that frequency and moreover, as FastProbe proceeds, it computes a more accurate value of $a_{ji}$ and $b_{ji}$. From the received signal strengths, it is also possible to compute the neighborhood set for $n_i$ (see Section III). Initially, the reputation value of each node is equal to the expected probability of a node being malicious. In the following discussion, $\overline{\rho}$ refers to the average of $\rho_i$ over all SUs that are associated with the SBS.

### D. Detailed Description

In this section, we will describe FastProbe (see Algorithm 1) in detail. FastProbe is invoked by SBS regularly so that those SUs can also be detected that may *become malicious* later. SBS first selects a channel $c_j$ that is free over which the testing is

to be performed[1]. FastProbe then proceeds in multiple time slots and in each slot a subset of nodes is tested. Since the SBS has high confidence about the behavior of those SUs that have either very high or very low $\rho_i$, therefore SBS tests (Line 4) only those nodes (set $A$) whose reputation value lies in a certain range. With the help of $ComputeSchedule$ (Algorithm 3), the minimum length testing schedule is computed (Line 5). The SBS then starts the testing schedule such that in each slot $t$, SUs in set $B_t$ transmit PUE signals at the power level specified in $S$, while SUs in set $A_t$ sense (Lines 8-11). After each slot, using $TestTransmitter$ (Algorithm 2) SBS computes (Line 14) the probability ($p_i$) of each node (say $n_i$) in $B_t$ transmitting at the specified (or expected) power level ($T_i^e$). For each node $n_i$ in $B_t$, if $p_i$ is less than the threshold $\beta$ (Lines 15-16), then the transmission is said to be malicious and in that case SBS does not update the reputation value of SUs that were tested by $n_i$ in this slot.

Otherwise, FastProbe proceeds to check every node that was tested by $n_i$ in this slot. Also, for every such node $n_j$, the expected path loss at frequency $f$ for transmission from $n_i$ to $n_j$ (denoted by $P_{ij}^e$), can be computed using (4). The values of observed path loss ($P_{ij}$) may be different from the expected path loss ($P_{ij}^e$) due to either the noise associated with this transmission or because $n_j$ is maliciously reporting incorrect readings. Since noise follows a Gaussian distribution (see Section III), the scaled probability of noise strength being $\delta_{ij}$ can be computed (Lines 21-22) using the *Scaled Gaussian distribution* (see Section III), such that the probability is 1 when the noise strength is 0. Since $p$ is scaled, therefore this is also the probability of $n_j$ reporting correct sensing result to the SBS.

On the basis of the computed probability, reputation value of $n_j$ is updated (Lines 23-26). First, FastProbe checks if both the expected received power level at $n_j$ and the power level reported by $n_j$ are below or equal to noise floor. This case increases our confidence that $n_j$ was not acting maliciously during this test. Thus, FastProbe updates the reputation value of $n_j$ (Lines 23-24) with a reputation value of 1.0 for this test. However, if only at most one of the two values is below the noise floor (Lines 25-26), then FastProbe updates $\rho_j$ using Exponential Moving Average (EMA) of rate $\alpha$ and value $p$. Finally, the observed path loss $P_{ij}$ from $n_i$ to $n_j$ at frequency $f$ is added to the set of measurements, and $a_{ij}$ and $b_{ij}$ are also updated using *weighted linear least squares method* with higher weight given to recent readings (Lines 27-28).

For the rest of the nodes that were not tested by the SBS, the algorithm updates (Lines 30-31) their reputation value such that it approaches $\overline{\rho}$. This ensures that all SUs will be tested repeatedly, so that the time variant behavior of SUs can be captured.

To reduce the amount of measurements maintained, an upper limit can be set on number of measurements that can be maintained for every pair of SUs. SBS will then need to discard the old measurements as it adds new ones (Line 26). This also allows FastProbe to capture long term variations that occur in path loss *e.g.* morning vs evening variation.

[1]Such a channel can be obtained by requiring the SUs that are currently labeled as non-malicious to scan the channel and then aggregating their results.

**Algorithm 1:** FastProbe

1 **Input**: $\rho_i$ of all SUs, Neighborhood Graph $\mathcal{G}$
2 **Output**: Updated $\rho_i$ of all SUs
3 Select a channel $c_j$ that is free; $f \leftarrow$ Central frequency of $c_j$
4 $A \leftarrow \{n_i : \tau_l \leq \rho_i \leq \tau_h\}$
`// Testing Schedule`
5 $S \leftarrow ComputeSchedule(\mathcal{G}, A, \rho_i$ for all SUs$)$
6 **foreach** *slot $t$ in schedule $S$* **do**
7 $S_t \leftarrow$ Set of tests in slot $t$
8 $A_t \leftarrow$ Nodes to be tested in slot $t$
9 $B_t \leftarrow$ Set of testing nodes in slot $t$
10 Request nodes in $B_t$ to transmit at power level specified in $S_t$
11 Request nodes in $A_t$ to report their sensing data
12 **foreach** $n_i \in B_t$ **do**
13 $T_i^e \leftarrow$ Expected transmission power level of $n_i$ (as specified in $S_t$)
14 $p_i \leftarrow TestTransmitter(\rho_i, \mathcal{G}, \ldots)$
15 **if** $p_i < \beta$ **then**
16 **continue**
17 **foreach** $n_j : n_j \in A_t \cap N(n_i)$ **do**
18 $R_j \leftarrow$ Received power level reported by $n_j$
`// Observed path loss from` $n_i$ `to` $n_j$
19 $P_{ij} \leftarrow T_i^e - R_j$
`// Expected path loss from` $n_i$ `to` $n_j$
20 $P_{ij}^e \leftarrow a_{ij} \log f + b_{ij}$
21 $\delta_{ij} \leftarrow P_{ij} - P_{ij}^e$
`// Probability that difference is because of thermal noise`
22 $p \leftarrow e^{\frac{-\delta_{ij}^2}{2\sigma^2}}$
23 **if** $P_{ij} \leq \psi_j^+$ **and** $P_{ij}^e \leq \psi_j^+$ **then**
24 $\rho_j \leftarrow \alpha\rho_j + (1-\alpha)(1.0)$
25 **else**
26 $\rho_j \leftarrow \alpha\rho_j + (1-\alpha)p$
27 Add $P_{ij}$ to the set of measurements
28 Update $a_{ij}$ and $b_{ij}$ using Weighted Linear Least Square Method
29 **foreach** *node $n_i \in V \backslash A$* **do**
`// Exponential decay to` $\overline{\rho}$
30 $\rho_i \leftarrow \gamma\rho_i + (1-\gamma)\overline{\rho}$

**Algorithm 2:** TestTransmitter

1 **Input**: $\rho_i, \mathcal{G}, \ldots$
2 **Output**: Probability $p_i$ that $n_i$'s transmission was not malicious
3 $V_i \leftarrow \phi$ // Set of voting SUs
4 **foreach** $n_j : n_j \in A_t \cap N(n_i) \wedge \rho_j \geq \overline{\rho}$ **do**
5 $P_{ij}^e \leftarrow a_{ij} \log f + b_{ij}$ //Expected path loss for this transmission
6 $R_j^e \leftarrow T_i^e - P_{ij}^e$//Expected received signal strength at $n_j$
7 **if** $R_j^e > \psi^+$ **then**
8 $V_i \leftarrow V_i \cup \{n_j\}$
`// Transmission power level of` $n_i$ `as observed by` $n_j$
9 $L_j \leftarrow R_j + P_{ij}^e$
`// Observed transmission power of` $n_i$
10 $T_i \leftarrow \frac{\sum_{n_j \in V_i} \rho_j L_j}{\sum_{n_j \in V_i} \rho_j}$
11 $\delta_i \leftarrow T_i^e - T_i$; $p_i \leftarrow e^{\frac{-\delta_i^2}{2\sigma^2}}$
12 **return** $p_i$

*Computing the probability of transmission being not malicious:* To compute the probability of $n_i$ transmitting at the specified (or expected) power level ($T_i^e$), Algorithm 2 employs a voting procedure where those neighbors of $n_i$ participate (Lines 3-8 of Algorithm 2) whose reputation values are high and for whom the SBS expects the received signal strength ($R_j^e$) to be more than $\psi^+$. Using the received signal strength reports from each of these SUs, SBS determines (Line 9) the transmission power level of $n_i$ as observed by $n_j$ (denoted by $L_j$). Finally, SBS computes the observed transmission power level of $n_i$ by taking a weighted average of observed transmission powers at SUs (Line 10). The observed transmission power may be different from $T_i^e$ due to either the thermal noise or because $n_i$'s transmission was malicious. $TestTransmitter$ then computes (Line 11) the scaled probability $p_i$ of this difference being due to thermal noise using a *Scaled Gaussian Distribution* such that the probability is 1 when $T_i$ and $T_i^e$ are equal. Since $p_i$ is scaled, therefore $p_i$ is also the probability of this transmission being not malicious and hence, value of $p_i$ is returned back to FastProbe.

*Computing the Testing Schedule:* For a given set $A$ of nodes to be tested, the testing schedule $S$ must satisfy the following requirements: (i) $S$ must be *error-free* and every node in $A$ must be tested at least once in $S$; (ii) In order to reduce the number of malicious transmissions, nodes with high reputation value should be given priority when selecting testing nodes; (iii) To detect malicious nodes that report incorrect results only when the actual received power level is close to the noise floor and to prevent malicious SUs from distinguishing between PUE signals and the real PU signals, it is beneficial to test SUs at various values of received power levels; and, (iv) Length of $S$ should be small (see Section IV-B). This problem of computing the minimum length schedule can be seen as a combination of Minimum Dominating Set problem as well as Edge Coloring problem and lies in the domain of NP-Hard problems as proven in the Technical Report [17]. Intuitively, $n_i$ has a higher chance of being selected as a *tester* than $n_j$ if (i) $n_i$ has higher reputation value than $n_j$; and, (ii) $n_i$ can test more nodes than $n_j$. Specifically, when $n_i$ is selected for testing, the *benefit* added by it to $S$ is the product of its reputation value and the number of nodes that it can test.

Algorithm 3 proceeds to calculate the schedule $S$ by greedily fitting in as many transmissions as possible in each time slot on the basis of benefit added by them. A node is added as a *testing node* only if its transmission does not interfere with any of the existing transmissions in the same slot, thus making $S$ *error-free*. Further, to reduce the number of malicious transmissions, only those SUs can be added whose reputation value is high (Line 5). Finally, the slot number is incremented when more transmissions cannot be added without interfering with the transmissions in the previous slot (Lines 12-13).

**Algorithm 3:** ComputeSchedule

```
1  Input: G, A, ρ_i for all SUs
2  Output: Testing Schedule S
3  t ← 1, S ← φ
4  while A ≠ φ do
5      w_max ← 0, Y_max ← φ, B ← {n_j : ρ_j ≥ ρ̄}
6      foreach n_j ∈ B : Transmission from n_j at its highest
       power level does not interfere with existing transmissions
       in slot t do
7          T_j ← A random feasible power level
8          Y_j ← {n_k : n_k ∈ A ∩ N(n_j)}
9          w_j ← ρ_j × |Y_j|
10         if w_j > w_max then
11             Y_max ← Y_j, w_max ← w_j, n_max ← n_j,
               T_max ← T_j
12     if Y_max = φ then
13         t++
14     else
15         A ← A\Y_max
16         foreach n_i ∈ Y_max do
17             S ← S ∪ (n_i, n_max, T_max, t)
18 return S
```

## V. Discussion

In this section, we will discuss how FastProbe can be further extended to make it more robust and accurate.

*1) Testing by SBS:* Apart from testing SUs by neighboring nodes, SBS can also *infrequently* test nodes at varying power levels. This helps in 2 cases: (i) To test a SU that does not have any neighbors with sufficiently high $\rho$; and, (ii) To detect collusion of multiple neighboring SUs. Since, in our algorithm a SU being tested does not know who is transmitting, therefore, a malicious SU would also report incorrect received signal strengths when it is tested by SBS. Note that the base stations generally maintain path loss to/from every client in order to adjust data rates. Therefore, SBS can detect malicious SUs by comparing the path loss during a sensing test with the path loss computed from recent data communications. Further, if a node with high reputation value is repeatedly found to act maliciously when tested by SBS, then this indicates that this node along with its neighboring SUs are colluding.

*2) In-band sensing:* A similar algorithm can be used to test malicious nodes that do not perform in-band sensing. If SBS wants to check if $n_j$ is in-band sensing on channel $c_k$, then while $n_j$ is transmitting on $c_k$, SBS will ask a neighbor of $n_j$ to transmit PUE signals on $c_k$ for 2 seconds. FCC regulations require SUs to vacate the channel within 2 seconds of the arrival of the Primary User. Thus, if $n_j$ is still transmitting after 2 seconds, it indicates that $n_j$ is not faithfully performing the in-band sensing.

*3) Consistently malicious nodes:* Observe that a malicious node (say $n_j$) can always perform sensing but either increase or decrease the reported received power consistently by the same fixed amount for all sensing requests. In this case, because of the absence of the knowledge of the actual path loss among neighbors, SBS may not detect that $n_j$ is malicious. However, such a malicious SU can be detected as follows: If $n_j$ always reports increased power level, then consider the case when a neighbor $n_i$ of $n_j$ transmits at a power level such that the actual received power level at $n_j$ is below its noise floor but the expected power level at $n_j$ is above its noise floor. Therefore, $n_j$ would not detect $n_i$'s transmission and to SBS it will report the received power level to be $\psi$. However, then the path loss observed by SBS for this transmission would be more than $P_{ij}^e$ and hence reputation value of $n_j$ would be reduced. On the other hand, if $n_j$ always reports decreased power level, then it will be detected when SBS performs the testing (see

Section V-1). This is because it is expected that $n_j$ would not report lower received signal strength to the SBS for the regular data transmissions since SBS may then reduce the data rate to compensate for higher path loss, leading to loss in throughput for $n_j$.

*4) Distinguishing Sensing Tests from Real PU Transmissions:* FastProbe utilizes PUE signals to detect malicious SUs. In FastProbe, it is not possible for the malicious users to distinguish emulated signals from actual PU signals using the algorithms for PU Emulation Attack described previously in literature [19], [25], [26] because of the following reasons: (i) Presence of obstacles, unpredictable path loss on the wireless and unknown transmission power level makes it difficult for the malicious users to localize the transmitter; (ii) Multiple SUs may transmit sensing tests in the same time slot making it difficult for colluding malicious users to perform transmitter localization; and, (iii) Even if localization is performed, it may not be useful if the location of actual PUs is unavailable or if the PUs are mobile (*e.g.* microphones, public safety vehicles).

*5) Mobility:* Some of the SUs in the system may move and thus the path loss values maintained for them may not be correct. Therefore, for the SUs that have moved recently, their path loss information gets updated through weighted linear least squares method with higher weight given to recent readings (see Section IV-C).

*6) Overhead:* It may seem that sensing tests may induce high overhead in FastProbe compared to existing algorithms. However, the use of sensing tests in FastProbe allows the SBS to select only a small subset of SUs while doing the actual channel sensing. This is because in FastProbe, the SBS can proactively determine if a node is malicious or not. This is in contrast to the existing algorithms that passively identify the malicious users and thus, must select a high number of SUs for actual channel sensing so as to accurately determine the channel state. Further, in FastProbe, the SBS is aware of the ground truth (*e.g.* transmission power level, path loss between the testing node and the tested node). Thus, it can determine the malicious SUs with lower overheads and higher accuracy.

## VI. Performance Evaluation

### A. Simulations

Using a custom-built simulator, we evaluated the performance of FastProbe and compared it with state of the art solutions. In the simulation, we deployed varying number of SUs in a field of size 100 kms $\times$ 100 kms as SBS may have a range of as much as 100 km [4]. Table II gives the default values of various parameters used during the simulation. Each simulation was performed 100 times and the average along with 95% confidence interval is plotted. In order to avoid testing SUs when not required, the SBS initiated testing schedule through FastProbe only after size of the set $A$ was at least 20% of the total number of SUs associated with SBS. The mobility pattern of the SUs was simulated using data from [27]. Obstacles were deployed in the network and the slow fading due to obstacles was modeled [2]. To account for the fast fading due to thermal noise, a Gaussian noise with zero mean and variance of 1 dB [1] was also introduced on all paths. We assumed that sensing by energy detection takes 10 ms time since each sensing takes 1 ms [1] and every SU took mean of 10 readings to increase the time diversity (see Section III). Each simulation ran for a virtual duration of 15 minutes.

TABLE II. Default Simulation Parameters

| **Parameter** | **Value** |
|---|---|
| Field Size | 100 kms $\times$ 100 kms |
| Number of SUs | 400 |
| Number of malicious SUs | 80 |
| SU Range | 10 kms |
| Number of PUs | 40 |
| $\alpha, \beta$ | 0.9 |
| Number of Channels | 50 |

We compared the performance of FastProbe with that of Attack-tolerant Distributed Sensing Protocol (ADSP) [10] that arranges nodes in clusters. A node is marked as malicious if its actual reading differs from its expected reading computed using the shadow correlation model and the readings of its neighbors. Since most of the other algorithms (see Section II) are based on similar assumptions, therefore, we expect them to behave similarly. To account for the time variant behavior of PUs, on an average SBS sensed every channel once every two minutes. For cooperative sensing of a channel, SBS in FastProbe used a set of dominating subset of SUs while in ADSP all SUs were used for sensing [10]. FastProbe can perform accurate cooperate sensing with a smaller set of SUs since the SBS eliminates the malicious SUs proactively (See Section V-6 for details).

We simulated two different attack models: (i) *Attack 1:* Malicious nodes sense the channel but instead of reporting the correct power level to the SBS, they either report power level higher by 20%, lower by 20% or the correct power level, all with equal probability; and, (ii) *Attack 2:* Multiple malicious SUs located close to each other collude so as to improve the reputation value of one of the malicious nodes. In all the attack models, malicious SUs started behaving maliciously at random times within first two minutes of simulation and alternated between malicious and non-malicious behavior each lasting for three minutes.

We studied different parameters: (i) *Number of transmissions:* It counts the number of tests performed by FastProbe over the period of 15 minutes; (ii) *Fractional throughput loss per user:* We define this as the average fraction of time spent by SUs as well as the SBS in sensing, transmitting or receiving data related to sensing, and waiting to find a free channel to switch to when PU arrives on an in-band channel; and, (iii) *Sensing Accuracy (or scanning accuracy):* Percentage of times the state of a channel was correctly determined (occupied by PU vs. not occupied). Thus, sensing accuracy also penalizes the algorithms for false positives.

*1) With variation in number of SUs:* In these set of simulations the density of SUs was increased while keeping the percentage of malicious SUs fixed at 20%. As the density of SUs increases, more SUs have to be tested resulting in total number of transmissions to increase (Figure 2(a)). However, higher density also enables a single SU to simultaneously test more neighbors. Thus, the number of transmission tests in FastProbe increases at the beginning. However, it becomes fairly constant after some time.

In terms of throughput loss per user, for both FastProbe and ADSP, the throughput loss per SU decreases with increase in number of SUs (Figure 2(b)). For both the algorithms, the increase in number of SUs helps in more accurate identification of malicious users. Thus, the sensing accuracy increases for both the algorithms resulting in SUs spending shorter time in waiting to find a free channel. For FastProbe, the decrease in number of transmissions Figure 2(a)) further helps in reducing the throughput loss. On an average, the throughput loss per user for ADSP remains at least $2x$ than that of FastProbe.

Accuracy of both the algorithms (Figure 2(c)) improve as the density increases since more SUs result in better coverage of the area. However, due to the presence of obstacles in the system and absence of ground truth, ADSP still has lower sensing accuracy compared to FastProbe.

*2) Variation in churn rate:* In a wireless system, continuous arrival and departure of SUs may happen due to SUs moving in/out of transmission range of SBS, network card of SUs going to sleep mode to save energy, or users turning off their devices. Figure 3 shows the performance results of FastProbe and ADSP under continuous churn of users with varying rates. Since ground truth is known in FastProbe, therefore, it is able to determine faster if a newly arrived node is malicious or not. On the other hand, ADSP takes multiple rounds to determine the accuracy of a node. Thus, faster churn rates causes comparatively higher throughput and accuracy loss for ADSP when compared to FastProbe.

### B. Experiments

To evaluate the performance of FastProbe in real networks and to compare it with ADSP [10], we deployed FastProbe and ADSP on 9 laptops that acted as SUs and SBS. The laptops operated on total of 5 channels in the 2.4Ghz and 5Ghz ISM bands using 802.11a/b protocol. 3 PUs were also deployed in the network. SUs were arranged in two clusters as shown in Figure 4 such that it was possible for SBS to communicate with all SUs and vice-versa. Presence of two walls affected the correlation among some of the neighboring SUs. Number of malicious SUs varied from 1 to 5 while the set of SUs that acted maliciously changed randomly with every iteration of the algorithm. We evaluated three parameters: (i) *Fractional throughput loss per user* (ii) *Sensing accuracy* (iii) *Average time taken to detect malicious SUs:* For this, we set a threshold ($\omega$) of 0.7 such that if the reputation value of an SU drops below it, then the SU is assumed to be *detected* as malicious.

As we can see from Figure 5(a), FastProbe detected malicious SUs with as much as 65% less throughput loss while maintaining higher accuracy. With increase in number of malicious users, the accuracy of detection decreased for both algorithms (Figure 5(b)). With increasing number of malicious users, it becomes difficult for both algorithms to detect them. This increases the average detection latency for both the algorithms (Figure 5(a)).

## VII. Conclusion

Cooperative sensing can be used to increase sensing accuracy. However, presence of malicious SUs that do not perform sensing and report incorrect results leads to inaccuracies in cooperative sensing. In this paper, we described *FastProbe*, a novel active transmissions based algorithm for detecting such malicious SUs. FastProbe is the first algorithm that can proactively detect malicious SUs, thereby preventing Cognitive Radio Networks from making incorrect sensing decisions. This helps in reducing the interference to Primary Users as well as increases the accuracy of sensing. FastProbe is also the first algorithm to detect malicious SUs that do not perform in-band sensing. Finally, through simulation and experimental results, we showed that FastProbe reduces the throughput loss due to sensing by as much as 65% while achieving higher accuracy compared to existing algorithms.

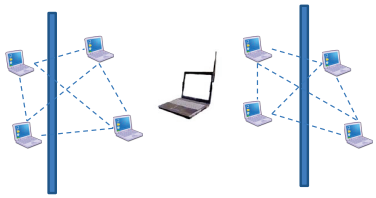

Fig. 4. Topology used for experiments: Links show neighborhood relationship. The SBS (at the center) is in the transmission range of all SUs. Two walls were used as obstacles.

## References

[1] H. Kim and K. G. Shin, "In-band Spectrum Sensing in Cognitive Radio Networks: Energy Detection or Feature Detection?" in *Proc. of ACM MobiCom*, 2008.

[2] S. Mishra, A. Sahai, and R. Brodersen, "Cooperative Sensing Among Cognitive Radios," in *Proc. of IEEE ICC*, 2006.

[3] P. Kaligineedi, M. Khabbazian, and V. Bhargava, "Malicious User Detection in a Cognitive Radio Cooperative Sensing System," *IEEE TWC*, vol. 9, no. 8, pp. 2488–2497, 2010.

[4] O. Fatemieh, A. Farhadi, R. Chandra, and C. Gunter, "Using Classification to Protect the Integrity of Spectrum Measurements in White Space Networks," in *Proc. of NDSS*, 2011.

[5] N. Kothari *et al.*, "Finding Protocol Manipulation Attacks," in *Proc. of ACM SIGCOMM*, 2011.

[6] Wikipedia, "Hacking of Consumer Electronics," http://en.wikipedia.org/wiki/Hacking_of_consumer_electronics.

[7] A. W. Min, K.-H. Kim, and K. G. Shin, "Robust Cooperative Sensing via State Estimation in Cognitive Radio Networks," in *Proc. of IEEE DySPAN*, 2011.

[8] T. Bansal, B. Chen, and P. Sinha, "DISCERN: Cooperative Whitespace Scanning in Practical Environments," in *Proc. of IEEE INFOCOM*, 2013.

[9] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 2002.

[10] A. Min, K. Shin, and X. Hu, "Attack-Tolerant Distributed Sensing for Dynamic Spectrum Access Networks," in *Proc. of IEEE ICNP*, 2009.

[11] A. W. Min, K. G. Shin, and X. Hu, "Secure Cooperative Sensing in IEEE 802.22 WRANs Using Shadow Fading Correlation," *IEEE Transactions on Mobile Computing*, 2010.

[12] O. Fatemieh, R. Chandra, and C. Gunter, "Secure Collaborative Sensing for Crowd Sourcing Spectrum Data in White Space Networks," in *Proc. of IEEE DySPAN*, 2010.

[13] H. Li and Z. Han, "Catch Me If You Can: An Abnormality Detection Approach for Collaborative Spectrum Sensing in Cognitive Radio Networks," *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, pp. 3554–3565, 2010.

[14] W. Wang, H. Li, Y. Sun, and Z. Han, "Securing Collaborative Spectrum Sensing Against Untrustworthy Secondary Users in Cognitive Radio Networks," *EURASIP Journal on Advances in Signal Processing*, 2010.

[15] K. Zeng, P. Paweczak, and D. Cabric, "Reputation-based Cooperative Spectrum Sensing with Trusted Nodes Assistance," *IEEE Communications Letters*, vol. 14, no. 3, pp. 226–228, 2010.

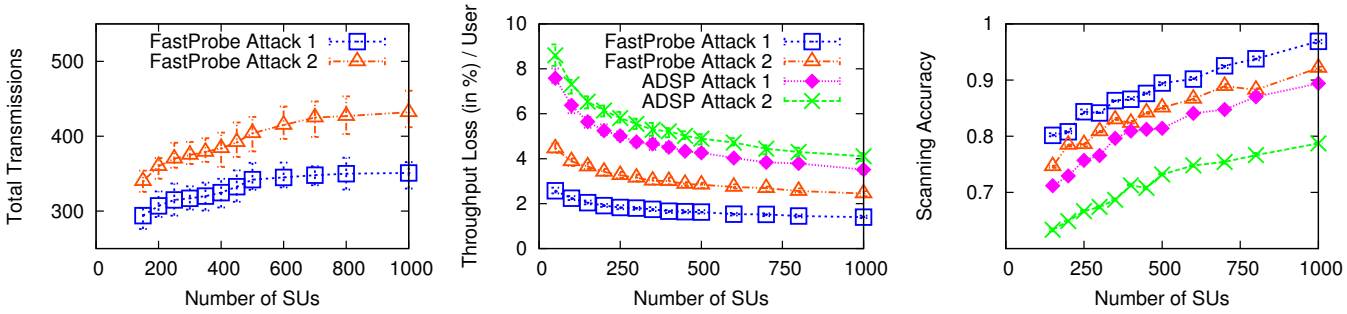


Fig. 2. (a): Variation in total number of test transmissions in FastProbe with varying number of SUs under two different attack models. (b)-(c): Comparison of FastProbe with ADSP [10].

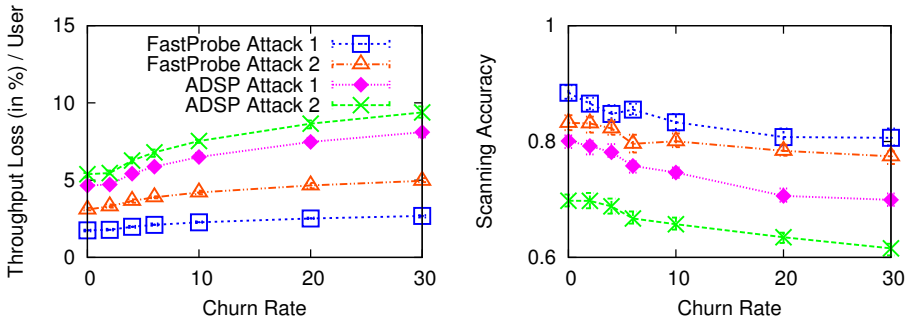


Fig. 3. Variation in throughput loss per user and sensing accuracy for increasing churn rate. A churn rate of 10 implies that on an average, in a window of 15 seconds, 10 users joined the network while 10 users left the network. Average number of SUs in network were 400.

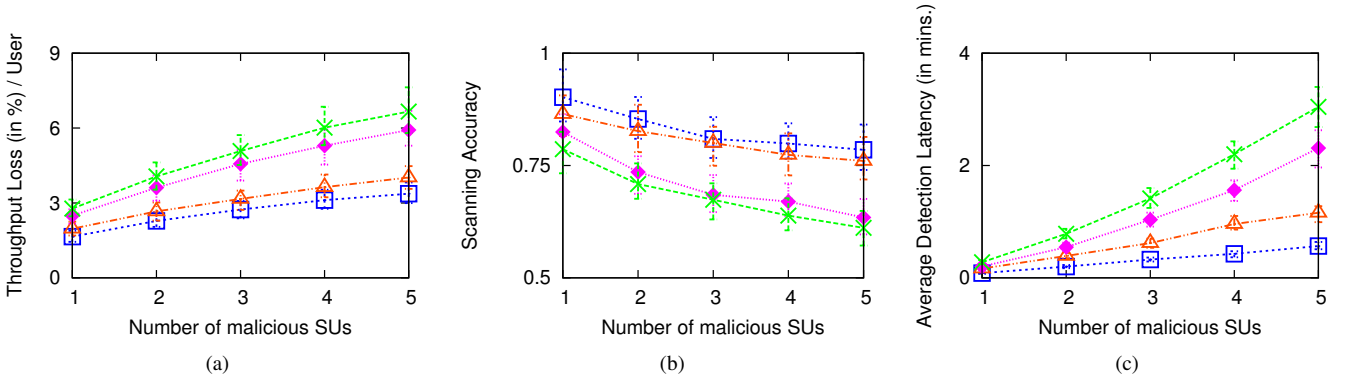


Fig. 5. (a) - (c): Experiment Results with variation in number of malicious SUs.

[16] A. Rawat, P. Anand, H. Chen, and P. Varshney, "Collaborative Spectrum Sensing in the Presence of Byzantine Attacks in Cognitive Radio Networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 2, pp. 774–786, 2011.

[17] T. Bansal, B. Chen, and P. Sinha, "Malicious User Detection in Cognitive Radio Networks Through Active Transmissions," Tech. Rep., http://www.cse.ohio-state.edu/~bansal/FastProbeTechRep.pdf.

[18] T. Newman and T. Clancy, "Security Threats to Cognitive Radio Signal Classifiers," in *Virginia Tech Wireless Personal Communications Symposium*, 2009.

[19] S. Liu, L. Greenstein, W. Trappe, and Y. Chen, "Detecting Anomalous Spectrum Usage in Dynamic Spectrum Access Networks," *Ad Hoc Networks*, 2011.

[20] S. Shellhammer *et al.*, "Performance of Power Detector Sensors of DTV Signals in IEEE 802.22 WRANs," in *Proc. of TAPAS*, 2006.

[21] A. Min and K. Shin, "An Optimal Sensing Framework Based on Spatial RSS-profile in Cognitive Radio Networks," in *Proc. of SECON*, 2009.

[22] R. Tandra and A. Sahai, "SNR Walls for Feature Detectors," in *Proc. of IEEE DySPAN*, 2007.

[23] M. Hata, "Empirical Formula for Propagation Loss in Land Mobile Radio Services," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 3, pp. 317–325, 1980.

[24] L. Sang, A. Arora, and H. Zhang, "On Exploiting Asymmetric Wireless Links via One-way Estimation," in *Proc. of MobiHoc*, 2007.

[25] S. Liu, Y. Chen, W. Trappe, and L. Greenstein, "ALDO: An Anomaly Detection Framework for Dynamic Spectrum Access Networks," in *Proc. of IEEE INFOCOM*, 2009.

[26] R. Chen, J. Park, and J. Reed, "Defense Against Primary User Emulation Attacks in Cognitive Radio Networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 1, pp. 25–37, 2008.

[27] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong, "CRAWDAD trace set ncsu/mobilitymodels/gps (v. 2009-07-23)," Downloaded from http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels/GPS, Jul. 2009.