Distributed Detection of Mobile Malicious Node Attacks in Wireless Sensor Networks

Jun-Won Ho*,[a], Matthew Wright[a], Sajal K. Das[a]

[a] *Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019-0015 USA*

Abstract

In wireless sensor networks, sensor nodes are usually fixed to their locations after deployment. However, an attacker who compromises a subset of the nodes does not need to abide by the same limitation. If the attacker moves his compromised nodes to multiple locations in the network, such as by employing simple robotic platforms or moving the nodes by hand, he can evade schemes that attempt to use location to find the source of attacks. In performing DDoS and false data injection attacks, he takes advantage of diversifying the attack paths with mobile malicious nodes to prevent network-level defenses. For attacks that disrupt or undermine network protocols like routing and clustering, moving the misbehaving nodes prevents them from being easily identified and blocked. Thus, mobile malicious node attacks are very dangerous and need to be detected as soon as possible to minimize the damage they can cause. In this paper, we are the first to identify the problem of mobile malicious node attacks, and we describe the limitations of various naive measures that might be used to stop them. To overcome these limitations, we propose a scheme for distributed detection of mobile malicious node attacks in static sensor networks. The key idea of this scheme is to apply sequential hypothesis testing to discover nodes that are silent for unusually many time periods—such nodes are likely to be moving—and block them from communicating. By performing all detection

*Corresponding author

and blocking locally, we keep energy consumption overhead to a minimum and keep the cost of false positives low. Through analysis and simulation, we show that our proposed scheme achieves fast, effective, and robust mobile malicious node detection capability with reasonable overhead.
Key words: Mobile malicious node, intrusion detection, sequential hypothesis testing

---

1. Introduction

Security researchers have long recognized that wireless sensor networks, with low-resource nodes that are typically left unattended and meant to be self-organizing, are potentially vulnerable to a wide range of attacks. Cryptographic protocols can prevent some attacks, such as by authenticating packets at each hop to ensure that they originate from legitimate nodes. However, an attacker can still physically capture sensor nodes, extract their cryptographic keying material, and modify their code to behave maliciously [11]. He can also remotely compromise nodes by injecting malicious code via the exploit of certain types of software vulnerabilities [7, 10]. With a small subset of nodes compromised through either approach, the attacker can launch a number of damaging attacks. For example, he can corrupt the monitoring operation of the base station by injecting false sensor readings, slow or stop the network by performing a distributed denial of service (DDoS) attack, or undermine critical operations such as routing, cluster formation, time synchronization, and localization. Thus, it is very important to quickly detect and stop malicious sensor nodes to protect the sensor network and the integrity of its operations.

Prior work has investigated the detection of malicious nodes through techniques such as traceback [21, 27] and local misbehavior detection [2, 9, 18, 20]. These approaches are based on the assumption that the malicious activity will come from a set of fixed locations to which we can narrow our search. This assumption generally holds in static sensor networks, in which the sensor nodes do not move. This assumption can be undermined, however, by *mobile mali-*

cious nodes. Although we only consider static sensor networks in this paper, the attacker could use a variety of methods to move his compromised nodes among the static nodes in the network. He could put the sensor nodes on small robots to create mobile sensors that move according to a pattern that the attacker chooses for the purpose of evading detection. Although such robotic platforms are becoming realistic and practical technology [3, 17], a less sophisticated attacker with modest manpower could periodically visit each node, pick it up, and move it. Both of these methods require some additional cost, but the attacker will often be willing to incur a greater cost to avoid losing the use of his compromised nodes when they are detected.

If the attacker can create mobile malicious nodes, he will benefit from the diversification of attack paths made possible by the nodes' movements. For instance, if the attacker wishes to launch a DDoS attack against the base station, he can employ a set of mobile malicious nodes that move to as many different locations as possible and flood the network from each location in turn. The flooding packets can be authenticated locally but will fail to authenticate at the base station, requiring some form of traceback to identify each attacker. In this attack scenario, however, malicious packets will traverse as many distinct paths as the number of locations that the mobile malicious nodes visit. Thus, traceback [21, 27] and other network-level defenses [5, 23] designed to find and stop immobile malicious nodes will fail.

If the attacker does not target the base station, but instead aims to defeat distributed self-organization protocols, such as routing and cluster formation, he can greatly raise the impact of compromising relatively few nodes by moving them around. A small group of mobile malicious nodes can disrupt these operations in one region at a time as they move around in the network, creating routing black holes, protocol failures due to time synchronization and localization errors, and other forms of havoc that otherwise would require many more compromised nodes to achieve. Local detection schemes, such as recently proposed schemes that use statistical methods to detect false data injection attacks [2, 18], would not be able to collect enough data to make a decision before

the mobile malicious nodes had moved away. While this also means that the region has time to recover, it will face periods of instability as the malicious nodes leave and then return to disrupt operations again. Hence, mobile malicious node attacks will have a much worse impact on the network than regular malicious node attacks.

1.1. Our Contributions

In this paper, we focus on the problem of detecting mobile malicious nodes in a static (immobile) wireless sensor network. To our knowledge, we are the first to address this problem. We begin (§2) with a model of the network and the attackers' capabilities using mobile malicious nodes. We point out that existing schemes for detecting misbehavior are necessary to stop attacks but are not sufficient to deal with mobile malicious nodes. We then describe (§3) the importance of detecting and stopping mobile malicious nodes and the drawbacks that simple, naive defenses present; other approaches either have high energy costs or can be undermined by a determined attacker.

To overcome these drawbacks, we propose (§4) a distributed detection scheme to identify and block mobile malicious nodes by leveraging the Sequential Probability Ratio Test (SPRT) [22]. Our scheme is designed to quickly detect and revoke mobile malicious nodes in a fully distributed manner. We leverage the intuition that immobile sensor nodes appear to be present around their neighbor nodes and communicate with them regularly. On the other hand, mobile malicious nodes are absent, and therefore silent, in locations in which they previously used to be. We describe how we embed this intuition into the SPRT (§4.1) so that neighbors can detect and block nodes that are silent unusually often. False positives, which are an issue with any detection scheme, here only serve to remove low-value communications links between nodes that seldom communicate with each other. Our approach requires no communication overhead and very reasonable storage and computational overheads at each node.

We validate the robustness and efficiency of our scheme through analysis and extensive simulation. Specifically, we prove that mobile malicious nodes are

quickly detected and revoked by our scheme (§4.3) with low false negative and false positive error rates (§4.2). Since the attacker could attempt to undermine our scheme by using limited movement patterns, we demonstrate that such an attacker would also have very limited success in his attacks and could be subject to traceback (§4.2). Moreover, we evaluate our scheme through simulation (§5). Our results show that our scheme detects mobile malicious nodes with only a few samples and very low false positive and negative rates.

2. Preliminaries

In this section, we describe the network assumptions and attacker model we use for our study.

2.1. Network Model

We investigate attacks and defenses in a static (*immobile*) sensor network in which sensor nodes are fixed to their locations after deployment. We also assume bidirectional communication links, such that any pair of nodes that can communicate can both send and receive from each other.

Moreover, we assume that sensor nodes are deployed in an *automated* manner. In an automated deployment, an aircraft or mobile robot randomly scatters many sensor nodes over the field. This assumption is common in sensor networks research, as it makes deployment inexpensive and manageable compared with manual deployment, especially for large networks with many nodes. Under this assumption, researchers have addressed a variety of security problems, including key distribution, replica detection, and secure localization [1, 4, 14, 29]. Since sensor nodes cannot know their neighbor nodes before deployment, there should be a *neighbor discovery period* after the initial deployment and after any redeployments, enabling each sensor node to find its neighbors.

Finally, we assume that the system is using some form of trackback [21, 27] or local misbehavior detection [2, 9, 18, 20]. Although not necessary for our scheme to detect mobile nodes, the attacker does not need to use mobile nodes if a scheme like this is not in place to block static malicious nodes.

2.2. Attacker Model

We assume that an attacker can create malicious nodes through physically capturing sensor nodes or remotely compromising them by exploiting software vulnerabilities. We also assume that he can make malicious nodes into mobile malicious nodes by moving them by hand or putting them on mobile platforms. As described in Section 2.1, every benign sensor node is assumed to be static (immobile) in the network. Thus, every mobile node is assumed to be malicious in static sensor networks. Although we do not place any limits on the movement strategy of the mobile malicious nodes, we observe two features of the compromise and movement strategy that are reasonable for maximizing the attacker's impact. First, mobile malicious nodes should visit as many locations as possible during the neighbor discovery period to make them be accepted as neighbors of as many nodes as possible. Second, the attacker can maximize his impact on many regions of the network and increase the diversity of the paths used by messages in his attacks by moving each node frequently.

The length of the neighbor discovery period is therefore a potential constraint on the attacker. Note, however, that the neighbor discovery period should be long enough for nodes to find their neighbors despite varying deployment times and clock skew. Also, it has been shown that nodes can be compromised in under one minute [11], thus leaving the attacker time to connect to multiple neighbors. In studying our defense, we conservatively assume that the attacker is not limited in the number of neighbors it may get for its mobile malicious node. After the neighbor discovery period, they start to launch attacks while freely moving within the regions in which they have been accepted as neighbors.

We do limit the attacker in one important way: we assume that mobile malicious nodes are equipped with the same wireless hardware as the original nodes. We note that an attacker can employ radios with very high signal strength to achieve a similar effect to moving nodes. However, with enough such radios, the attacker can also effectively be in multiple places at once. Rather than deal with this attacker type directly, we instead note that several researchers have proposed methods to detect the use of strong signal power [1, 6, 12]. The net-

work can use one of these schemes to detect malicious nodes that virtually move with strong signal power. Thus, our attacker is limited to physically moving the nodes.

Finally, we note that the attacker model only makes sense in a static sensor network. It may still be interesting a network in which most sensors are in fixed locations but communication is facilitated with a small set of mobile nodes, e.g. serving as data mules. In this case, we would need to develop additional mechanisms to cope with the compromise of one or more mobile nodes.

3. The Challenge Presented by Mobile Malicious Node Attacks

Before we describe our solution for mobile malicious node detection, we first point out the importance of investigating this problem. We begin by describing why using mobile malicious nodes is attractive and even essential for the attacker. Then, although we are the first to our knowledge to point out the issue of mobile malicious nodes, we note that other existing techniques could be tried to stop them. We point out the limitations of these other approaches in that they either cannot stop the attacker or cause other problems in the network.

3.1. Attacker Benefits

As noted in Section 1, an attacker benefits greatly from using mobile malicious node attacks, due to increased path diversity and the difficulty of local detection. We now briefly argue that attacks based on compromised nodes are important for many attacker goals, that immobile malicious node attacks are subject to detection techniques that would render the nodes useless, and that an attacker would want to use mobile malicious nodes to improve his compromised node attacks.

If an attacker wants to slow or stop the operations of a self-organizing wireless network, he essentially has two choices—he can either jam and disable nodes or compromise them. Dealing with jamming and disabling of nodes is a hard problem, but it leaves little doubt as to the presence of an attacker and even the range of his attacks. Jamming can be detected and even mapped out in

a sensor network [19, 24, 26]. Compromising nodes is a much more effective way for the attacker to eavesdrop on network activity, cause disruption, and introduce false sensor readings without as readily giving away the attacker's presence and activity.

Recent research, however, would detect and stop most immobile malicious node attacks. In particular, traceback techniques can be used to find the location of the malicious nodes given enough attack packets [21, 27]. This prevents attacks like DDoS. If the attacker instead chooses to inject false data or misbehave in the context of protocols like routing, the compromised nodes can be discovered through techniques for local misbehavior detection [2, 9, 18, 20]. Once detected, not only are the compromised nodes blocked, but the attacker's presence and activity are revealed to the network operator, and the attacker must compromise additional nodes to continue his attack.

Faced with this reality, mobile malicious node attacks should be quite attractive to the attacker. They would prevent existing detection techniques from identifying the malicious nodes, keep the network operator from fully measuring the extent and nature of the attack, and allow the nodes to remain part of the network.

Thus, to effectively combat a smart attacker, we must compliment existing techniques for detecting *immobile* malicious nodes with techniques for detecting and blocking mobile malicious nodes.

3.2. Limitations of Other Approaches

We believe that a mobile malicious node detection scheme should be fast and accurate in detection, have low overhead, and be effective in the face of determined attackers. All of the approaches we describe below do not meet this standard in one or more ways.

A simple approach to block mobile malicious nodes is to install the list of neighboring nodes into each sensor node in the pre-deployment stage and have each node reject communications with nodes that are not included in its list. This prevents mobile malicious nodes from communicating with additional

nodes or moving beyond a small area. This approach, however, requires much more work for the network operator in the pre-deployment stage, as the full topology of the network must be determined in advance. Moreover, since there are always deployment errors when using automatic deployment, it is highly likely that each node's actual neighbors are different from the pre-installed neighborhood list, causing many benign nodes to be rejected by their neighbors. It was shown in [14] that using this simple approach (their *Scheme I*) leads to substantial network connectivity problems.

Location distinction schemes [8, 15, 16, 28] might be tried to detect mobile malicious nodes. In these schemes, receivers discern location changes of senders in wireless networks based on received signal strength (RSSI) or temporal radio link signatures. However, the attacker can simply turn off the node's radio during movement and move nodes far enough each time to prevent any one receiver from hearing a malicious node sending packets from two different locations.

Another simple, but naive, approach to detecting mobile malicious nodes is for all nodes to report their neighbor lists to the base station. Although this approach quickly detects mobile malicious nodes, it has several limitations. First, it is a centralized approach that has sizable overheads due to each node sending these reports to the base station and sending updates whenever new nodes join the network. Second, it's detection capability can be undermined by the attacker if each mobile node only becomes neighbors with one node in each region. Third, the attacker can induce substantial false positives by sending the base station false reports that make immobile benign nodes seem like mobile malicious nodes.

Given that the approaches described in this section have major limitations, we believe it is very important to design a new approach for detecting and stopping mobile malicious nodes. In the rest of this paper, we show how we achieve this goal with a scheme that is fast, effective, accurate in detection, fully distributed, and very low in overhead.

4. Detecting Mobile Malicious Nodes Using the SPRT

In this section, we present the details of our scheme for mobile malicious node detection.

As described in Section 2.2, we regard any mobile node in a static sensor network as malicious. Based on this, our main intuition for mobile malicious node detection is that a mobile malicious node does not send any messages to its neighbors in a given region when it is elsewhere in the network. A simple way to leverage this intuition is to have each node u measure the time between observed messages for each neighbor node v (the *absence period*) and compare this time to a pre-defined threshold. If the absence period is more than the threshold value, node u decides that v has been absent from the area and is therefore a mobile malicious node. Node u then decides to no longer accept communications from node v. Note that by using entirely local detection and revocation, there are no negative consequences if v has simply lost power or if u's connection to v is weak and cannot be used reliably in any case. This simple approach efficiently and effectively detects mobile malicious nodes as long as the threshold value is properly configured.

However, it is not easy to configure a proper threshold value to detect mobile malicious nodes. If we set the threshold too high, it is likely that we get false negatives in which some mobile malicious nodes bypass detection. On the other hand, if we set the threshold too low, it is likely that we get many false positives in which benign nodes are detected as mobile malicious nodes and blocked from communicating. To minimize these errors, we need to set up the threshold in such a way that it is dynamically changed in accordance with the measured absence time duration for a node. To satisfy this need, we use the Sequential Probability Ratio Test (SPRT) [22], which is a statistical decision process and can be thought of as a dynamic threshold scheme [13]. In the SPRT, a random walk proceeds between upper and lower limits that are dynamically configured in accordance with the type of observation. The movement of random walk is dynamically set to conform with the type of observation. If the walk reaches or

exceeds the lower or upper limit, it terminates in the acceptance of the null or alternate hypothesis, respectively. The SPRT has the advantage that it reaches a decision with few samples while achieving low false positive and false negative rates [22]. We thus can use it to get fast, accurate detection of malicious mobile nodes.

We apply the SPRT to the mobile malicious node detection problem as follows. Every sensor node performs the SPRT on each of its neighbor nodes. Each time a node observes the presence (resp. absence) of a neighbor node within the node's communication range, it will facilitate the random walk to hit or cross the lower (resp. upper) limit and thus lead to the acceptance of the null (resp. alternate) hypothesis that a node has not moved (resp. has moved). Once a node decides that its neighbor node has moved, it marks the neighbor node as a mobile malicious node and no longer communicates with it.

We now describe the scheme and then present analyses of its security and performance.

4.1. Protocol Description

Before deployment, the network operator assigns a unique ID to every sensor node. He also pre-loads onto each sensor node secret keying materials that are required for pairwise key establishment; we can use any key pre-distribution techniques for sensor networks such as [4, 29]. To prevent source ID forgery, we use a hop-by-hop authentication mechanism with a secret key shared between each pair of neighboring nodes. Namely, every message is authenticated at every intermediate node with a secret key shared between the node and its predecessor along the message's route. Finally, each sensor node divides time into a series of time slots.

After the initial deployment and after any redeployments, every node u discovers a set of neighboring nodes $N(u)$ during a neighbor discovery period. After the neighbor discovery period, u checks in each time slot whether each neighbor $v \in N(u)$ is present within u's communication range. Specifically, node u measures in each time slot the number of messages sent to it by v and uses it

as a metric to decide whether v is present or absent. Let S_i denote the number of messages sent by v during the ith time slot, $i \geq 1$. Let G_i be a Bernoulli random variable that indicates whether v was silent during the ith time slot; G_i is defined as:

$$G_i = \begin{cases} 1 & \text{if } S_i = 0 \\ 0 & \text{if } S_i > 0 \end{cases} \quad (1)$$

The success probability φ of a Bernoulli distribution is defined as $\varphi = \Pr(G_i = 1) = 1 - \Pr(G_i = 0)$. If φ is less than or equal to a preset threshold φ', it is likely that node v has not moved. On the other hand, if $\varphi > \varphi'$, it is likely that node v has moved. The problem of deciding whether v has moved or not can be formulated as a hypothesis testing problem with null and alternate hypotheses of $\varphi \leq \varphi'$ and $\varphi > \varphi'$, respectively.

To best understand the possibilities for error, we reformulate the above hypothesis testing problem as one with null and alternate hypotheses of $\varphi \leq \varphi_0$ and $\varphi \geq \varphi_1$, respectively, such that $\varphi_0 < \varphi_1$. In this reformulated problem, a false positive error occurs when the alternate hypothesis is accepted even though $\varphi \leq \varphi_0$ and a false negative error occurs when the null hypothesis is accepted even though $\varphi \geq \varphi_1$. To reflect these two types of errors in the decision process, we define a user-configured false positive rate α' and false negative rate β' in such a way that the false positive and false negative rates are respectively bounded by α' and β'.

Now we describe how node u performs the SPRT to decide about node v from a set of n observed samples, where S_i is the i-th sample. We define H_0 as the null hypothesis that v has not moved and H_1 as the alternate hypothesis that v has moved. We then define L_n as the log-probability ratio on n samples, given as:

$$L_n = \ln \frac{\Pr(G_1, \ldots, G_n | H_1)}{\Pr(G_1, \ldots, G_n | H_0)}$$

Assume that G_i values are independent and identically distributed. Then L_n

can be rewritten as

$$L_n = \ln \frac{\prod_{i=1}^{n} \Pr(G_i|H_1)}{\prod_{i=1}^{n} \Pr(G_i|H_0)} = \sum_{i=1}^{n} \ln \frac{\Pr(G_i|H_1)}{\Pr(G_i|H_0)} \tag{2}$$

Let ψ_n denote the number of times that $G_i = 1$ in the n samples. Then we have $L_n = \psi_n \ln \frac{\varphi_1}{\varphi_0} + (n - \psi_n) \ln \frac{1-\varphi_1}{1-\varphi_0}$, where $\varphi_0 = \Pr(G_i = 1|H_0)$, $\varphi_1 = \Pr(G_i = 1|H_1)$. The rationale behind the configuration of φ_0 and φ_1 is as follows. φ_0 should be configured in accordance with the likelihood of the occurrence that an immobile benign node is determined to have moved because the messages sent by it are totally lost due to message collision or other transmission problems. Note that if the sender ID is observed, the node will be considered to have sent a packet, even if the rest of the packet is lost. φ_1 should be configured to consider the likelihood of the occurrence that a mobile malicious node is determined to have moved.

On the basis of the log-probability ratio L_n, the SPRT for H_0 against H_1 is given as follows:

- $L_n \leq \ln \frac{\beta'}{1-\alpha'}$: accept H_0 and terminate the test.
- $L_n \geq \ln \frac{1-\beta'}{\alpha'}$: accept H_1 and terminate the test.
- $\ln \frac{\beta'}{1-\alpha'} < L_n < \ln \frac{1-\beta'}{\alpha'}$: continue the test process with another observation.

We can rewrite the SPRT as follows:

- $\psi_n \leq \omega_0(n)$: accept H_0 and terminate the test.
- $\psi_n \geq \omega_1(n)$: accept H_1 and terminate the test
- $\omega_0(n) < \psi_n < \omega_1(n)$: continue the test process with another observation.

Where:

$$\omega_0(n) = \frac{\ln \frac{\beta'}{1-\alpha'} + n \ln \frac{1-\varphi_0}{1-\varphi_1}}{\ln \frac{\varphi_1}{\varphi_0} - \ln \frac{1-\varphi_1}{1-\varphi_0}}, \quad \omega_1(n) = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\varphi_0}{1-\varphi_1}}{\ln \frac{\varphi_1}{\varphi_0} - \ln \frac{1-\varphi_1}{1-\varphi_0}}$$

If u accepts H_0, it determines that v is an immobile benign node and repeats the SPRT. However, if u accepts H_1, it determines that v is a mobile malicious node and will not cease communicating with v.

4.2. Security Analysis

In this section, we will first present an analysis of the detection accuracy of our proposed scheme and then describe the limitations of mobile malicious node attacks under the presence of our proposed scheme. Finally, we will describe the effect of false positives on the system.

4.2.1. Detection Accuracy

In the SPRT, the false positive rate α and false negative rate β are defined as the probability that H_1 (resp. H_0) is accepted when H_0 (resp. H_1) is true. The upper bounds of α and β are computed as $\alpha \leq \frac{\alpha'}{1-\beta'}$ and $\beta \leq \frac{\beta'}{1-\alpha'}$ in [22]. The upper bound of the summation of α and β is given by $\alpha + \beta \leq \alpha' + \beta'$ [22]. Because β is the false negative probability, $(1 - \beta)$ is mobile malicious node detection probability of a single node. Thus, the lower bound on the detection probability of a mobile malicious node is given by $(1 - \beta) \geq \frac{1-\alpha'-\beta'}{1-\alpha'}$. We can infer from the above equations that low user-configured false positive and false negative rates will lead to a high probability of mobile malicious node detection with few benign nodes being blocked by their neighbors. In essence, the error rates can be set arbitrarily low at the cost of slower detection speeds. We later show (§5.2) that nodes can perform detection quickly with reasonable error rates.

4.2.2. Limitations of Mobile Malicious Node Attacks

We now investigate whether mobile malicious nodes could behave so as to avoid being detected by our scheme. When the attacker deploys his mobile malicious nodes, he first has them visit as many locations as possible during the neighbor discovery period to get as many benign nodes as possible to be their neighbors. The main rationale behind this is that attacker can take advantage of diversifying his attack paths and spreading his attack impact via many neighboring nodes. Assume that a mobile malicious node becomes neighbors with

nodes $u_1, u_2, \ldots, u_k$ during the neighbor discovery period. After the neighbor discovery period, a mobile malicious node first goes to the area near u_1 and sends attack packets into the network through u_1 while staying in u_1's vicinity. It then repeats this process while sequentially moving to nodes $u_2, u_3, \ldots, u_k$.

We now show that attacker gets little benefit from using node movement when our scheme is employed. We first consider the case that the malicious node contacts a benign node early on after the neighbor discovery period and then we consider the case that the malicious node waits for some time before this first contact. By showing that the mobile malicious node is detected in either case, we show that avoiding detection requires the attacker to contact each neighbor node early and remain in its vicinity for the substantial portion of the time it is active in the network. Thus, our detection scheme greatly limits the number of regions in which the malicious node can operate effectively.

Lemma 4.1. *Let us consider the n time slots of node u_i, $1 \leq i \leq k$. Let us denote by γ the fraction of the n time slots in which mobile malicious node v meets u_i and sends malicious packets to the network via u_i while staying within the vicinity of u_i. The mobile malicious node v is detected by node u_i under the conditions that $0 < \gamma \leq \gamma^* = 1 - \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\varphi_0}{1-\varphi_1}}{n(\ln \frac{\varphi_1}{\varphi_0} - \ln \frac{1-\varphi_1}{1-\varphi_0})}$ and $n > \frac{\ln \frac{1-\beta'}{\alpha'}}{\ln \frac{\varphi_1}{\varphi_0}}$.*

PROOF. Since node u_i communicates with v in γn time slots, the number of packets sent to u_i by v is measured to be greater than zero for the first γn out of n time slots. Hence, u_i sets $G_j = 0$ for all j such that $1 \leq j \leq \gamma n$ because the jth time slot corresponds to the jth sample in the SPRT. Since v leaves from u_i and moves to u_{i+1} after the first γn time slots, u_i sets $G_j = 1$ for all j such that $\gamma n < j \leq n$. Accordingly, u_i sets $\psi_n = (1-\gamma)n$. If $\psi_n = (1-\gamma)n \geq \omega_1(n)$ holds, then u_i terminates the SPRT in acceptance of H_1 on v. Since $\gamma > 0$ and $\omega_1(n) = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\varphi_0}{1-\varphi_1}}{\ln \frac{\varphi_1}{\varphi_0} - \ln \frac{1-\varphi_1}{1-\varphi_0}}$, u_i detects mobile malicious node v if $0 < \gamma \leq 1 - \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\varphi_0}{1-\varphi_1}}{n(\ln \frac{\varphi_1}{\varphi_0} - \ln \frac{1-\varphi_1}{1-\varphi_0})}$ and $n > \frac{\ln \frac{1-\beta'}{\alpha'}}{\ln \frac{\varphi_1}{\varphi_0}}$ holds.

We now investigate how n affects γ^* when $\alpha' = \beta' = 0.01$. For this study, we consider three settings of the lower and upper threshold values φ_0 and φ_1:

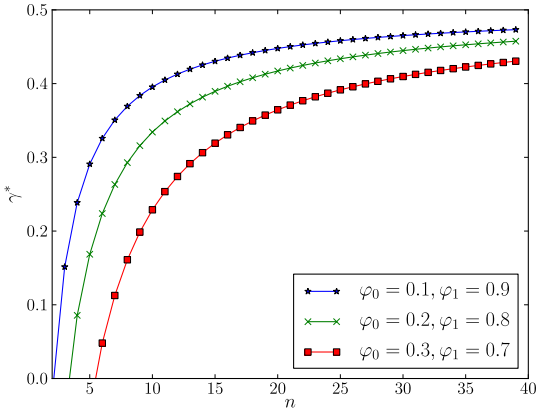

Figure 1: Minimum fraction of time slots in which the attacker needs to contact a given neighbor to avoid detection (γ^*) vs. number of time slots (n).

$\varphi_0 = 0.1$ and $\varphi_1 = 0.9$, $\varphi_0 = 0.2$ and $\varphi_1 = 0.8$, and $\varphi_0 = 0.3$ and $\varphi_1 = 0.7$. As shown in Figure 1, γ^* tends to increase as n rises in all three cases. However, the growth of γ^* slows such that the value of γ^* is sustained below 0.5. This indicates that node u_i detects mobile malicious node v as long as v stays within the vicinity of u_i for at most $\frac{n}{2}$ time slots out of n time slots.

Note that mobile malicious node v can bypass our detection scheme by staying within u_i's vicinity for more than half of the n time slots. However, this greatly reduces the benefits of having mobile malicious nodes. First, the attacker loses the most of the benefits of having highly diverse attack paths and attack amplification due to movement. The malicious node cannot even alternate between two neighbors u_1 and u_2 without being detected by one of them. Second, by being in any one region for such a high fraction of the time, the mobile malicious node must either greatly scale back its attack traffic or be subject to one of the existing mechanisms for static malicious node detection [2, 9, 18, 20, 21, 27]. Essentially, the attacker is stuck between using mobile nodes, which will be detected by our scheme, and using static nodes, which will be detected through existing approaches

A variation on this attack is to compromise a number of nodes, one for each

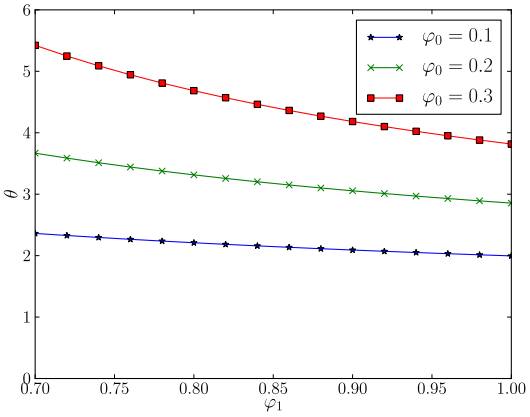

Figure 2: Maximum number of time slots after neighbor discovery period before contacting a given neighbor (θ) vs. the upper (detection) threshold (φ_1).

region, and have them take turns being the attacker. This makes them appear to be moving, though each node can send a message during each time slot so as to not be considered absent. This will evade our detection scheme, but the nodes remain subject to static malicious node detection [2, 9, 18, 20, 21, 27]. Detection time may increase in proportion to the amount that the attacker scales back his attack in each region.

Lemma 4.2. *Mobile malicious node v is detected as moving by node u_i, $1 \leq i \leq k$, if it does not meet u_i for the first n' time slots after the neighbor discovery period such that $n' \geq \theta = \frac{\ln \frac{1-\beta'}{\alpha'}}{\ln \frac{\varphi_1}{\varphi_0}}$.*

Proof. If v does not meet u_i for the first n' time slots after neighbor discovery period, u_i measures the number of packets sent to it by v as zero for the first n' time slots. Hence, u_1 sets $G_i = 1$ for all $1 \leq i \leq n'$. Accordingly, u_i sets $\psi_{n'} = n'$. If $\psi_{n'} = n' \geq \omega_1(n')$ holds, then u_i terminates the SPRT in acceptance of H_1 on v. Therefore, u_i detects mobile malicious node v if $n' \geq \frac{\ln \frac{1-\beta'}{\alpha'}}{\ln \frac{\varphi_1}{\varphi_0}}$ holds.

We now investigate how upper threshold φ_1 affects θ when $\alpha' = \beta' = 0.01$, and the lower threshold $\varphi_0 = 0.1, 0.2, 0.3$. As shown in Figure 2, θ decreases as φ_1 increases when φ_0 is fixed to 0.1, 0.2, and 0.3. On the other hand, θ

increases as φ_0 rises when φ_1 is fixed. We see from this that small values of φ_0 and large values of φ_1 lead to small values of θ. We also observe that θ is less than six in the above cases. This indicates that mobile malicious node v should meet node u_i within at most six time slots after neighbor discovery period in order to prevent it from being detected. Hence, our scheme will substantially limit the time period during which mobile malicious nodes are not detected.

By considering both Lemma 4.1 and Lemma 4.2, we can see that our scheme greatly limits attacker's gain from employing node movement. An increase in the time period during which mobile malicious node v interacts with node u_i will lead to an increase in the chance that v is detected by nodes $u_{i+1}, \ldots, u_k$ according to Lemma 4.2. On the other hand, if v interacts with u_i for a short time, it will be detected by u_i after leaving the vicinity of u_i and moving to u_{i+1} according to Lemma 4.1. Therefore, his attack will be ultimately detected irrespective of the movement strategies of his nodes.

Finally, we consider the *circular movement attack*. In this attack, attacker forms a circular list of nodes $u_1, u_2, \ldots, u_c$ and has a mobile malicious node sequentially contact the nodes in the circular list starting from u_1 while sending attack packets through $u_i, (1 \leq i \leq c)$. After meeting with u_c, mobile malicious node contacts u_1 again and repeats the above process. Attacker may expect to bypass our detection scheme by having the mobile malicious node move very quickly to spend time in all the regions, periodically communicate with u_i, and thus appear to be present in the vicinity of u_i. However, the mobile malicious node will be virtually regarded as static node after periodically communicating with nodes in the circular list and thus it will be ultimately detected by the existing mechanisms for static malicious node detection [2, 9, 18, 20, 21, 27]. Hence, the attacker will take little benefit from circular movement attack due to the existing static malicious node detection schemes.

4.2.3. False Positives

We now briefly examine the effect of false positives on the system. We know that using the SPRT will keep the false positive rate α at or below $\frac{\alpha'}{1-\beta'}$. Setting

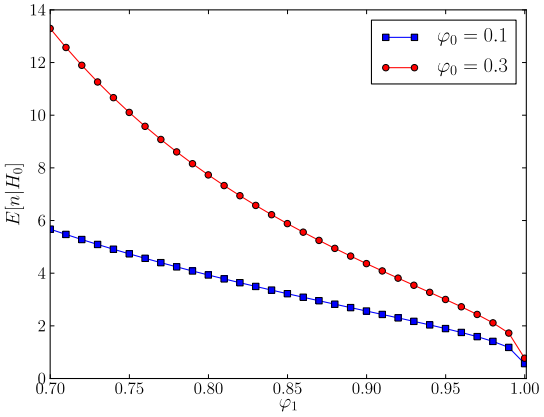

Figure 3: Ave. time slots to accept H_0 ($E[n|H_0]$) vs. the upper threshold (φ_1).

the user-configured false positive α' and user-configured false negative β' to a low value, such as 0.01, ensures that the number of nodes that are blocked by their neighbors will be low. We further note that, as the SPRT is a random walk, a node is likely to be detected by only one of its neighbors in a false positive detection case. Thus, it can still communicate with any of its other neighbors. On the other hand, the neighbors of a mobile malicious node will all correctly detect and block it eventually.

We also note that the main communication patterns in the system are the ones most likely to be maintained despite any false positives. If a node mainly communicates with a substantial subset of its neighbors, it would likely only be blocked by a small number of neighbors with whom it does not communicate regularly.

Finally, we consider the case in which sensor nodes will be misdetected as mobile malicious nodes if their batteries run out and thus they appear to be absent forever in the network. In this case, these nodes are actually in a dead state and thus they cannot participate in network operations. As a result, they will not be affected by the fact that they are incorrectly marked as mobile malicious nodes and blocked by their neighbors.

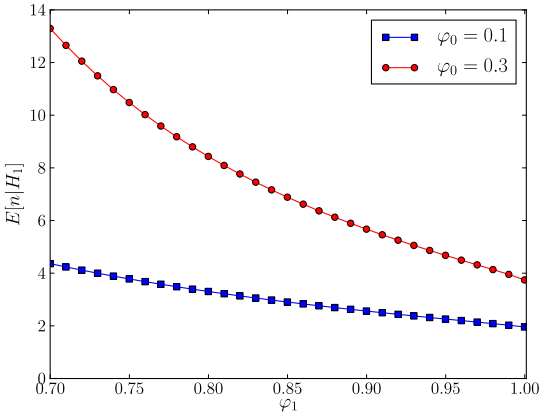

Figure 4: Ave. time slots to accept H_1 ($E[n|H_1]$) vs. the upper threshold (φ_1).

4.3. Performance Analysis

In this section, we compute how many samples are required on average for a node to decide whether its neighboring node is a mobile malicious node or not.

Let n denote the number of samples for the termination of the SPRT. Since n varies with the types of samples, it is thought of as a random variable with an expected value $E[n]$. According to [22], $E[n]$ is calculated as $E[n] = \frac{E[L_n]}{E\left[\ln \frac{\Pr(G_i|H_1)}{\Pr(G_i|H_0)}\right]}$. We then compute the expected values of n conditioned on hypotheses H_0 and H_1 as follows:

$$E[n|H_0] = \frac{(1-\alpha')\ln\frac{\beta'}{1-\alpha'} + \alpha'\ln\frac{1-\beta'}{\alpha'}}{\varphi_0\ln\frac{\varphi_1}{\varphi_0} + (1-\varphi_0)\ln\frac{1-\varphi_1}{1-\varphi_0}}$$

$$E[n|H_1] = \frac{\beta'\ln\frac{\beta'}{1-\alpha'} + (1-\beta')\ln\frac{1-\beta'}{\alpha'}}{\varphi_1\ln\frac{\varphi_1}{\varphi_0} + (1-\varphi_1)\ln\frac{1-\varphi_1}{1-\varphi_0}} \tag{3}$$

Figures 3 and 4 show how lower threshold φ_0 and upper threshold φ_1 affect $E[n|H_0]$ and $E[n|H_1]$, respectively. Specifically, $E[n|H_1]$ decreases as φ_1 rises when φ_0 is fixed to 0.1 and 0.3. This means that a large value of φ_1 contributes to detection of mobile malicious nodes quickly with few reports. For a given value of φ_1, $E[n|H_1]$ is less when $\varphi_0 = 0.1$ than when $\varphi_0 = 0.3$. This means that small values of φ_0 result in a smaller number of samples required for mobile

malicious node detection. In the case of $E[n|H_0]$, we see that a small value of φ_0 and a large value of φ_1 also lead to a decrease in the number of samples required to correctly decide that an immobile benign node is not moving.

5. Simulation Study

In this section, we will first describe our simulation environment and then discuss the simulation results.

5.1. Simulation Environment

We developed a simple simulation program to evaluate the proposed scheme in terms of its detection capability, accuracy, and speed.

In the simulation, we consider two cases. In the first case, a benign node u receives the packets from its immobile benign neighbor v. The number of packets generated by v is selected uniformly at random from the range $[0, N_{max}]$ in each time slot, where N_{max} is the maximum number of packets that can be generated by v. In the second case, mobile malicious node w contacts benign node u at the δth time slots after the neighbor discovery period. In case of $\delta = 0$, u is the first node that w contacts after neighbor discovery period. In case of $\delta > 0$, w meets other nodes for δ time slots after neighbor discovery period and thus it takes δ time slots from the end of the neighbor discovery period to the meeting between w and u. Since w is out of the vicinity of u for δ time slots, the number of messages sent by w would be zero for δ time slots and accordingly $\psi_\delta = \delta$ holds. We investigate how the values of δ affect the detection of mobile malicious node w. Note that $\psi_\delta = \delta \geq \omega_1(\delta)$ always holds when $\delta \geq 3$ and thus mobile malicious node w is always detected by u. Thus, we configure δ to 0 and 2. Moreover, benign node u receives packets generated by its mobile malicious neighbor w as long as w is in the vicinity of u. Once w moves out of the vicinity of u, u cannot receive any messages from w. We denote by τ the time period during which w stays within the vicinity of u and define a time slot as a unit of τ.

We also set the both the user-configured false positive threshold α' and the false negative threshold β' to 0.01, and we set lower and upper thresholds φ_0 and φ_1 to 0.1 and 0.9, respectively. We discuss the rationale behind these configurations in Section 4.3. To emulate packet loss due to packet collision, we modify the number of packets generated by a node with packet loss rate ρ. Specifically, we take the number of packets generated by a node (η), and generate η' selected uniformly at random from the range $[\eta - \eta\rho, \eta]$. This configuration reflects that the number of lost packets due to packet collision increases with greater values of ρ.

5.2. Simulation Results

We use the following metrics to evaluate the performance of our scheme:

- *Number of Time Slots* is the number of time slots required for a node to decide whether its neighboring node is a mobile malicious node or not.
- *False Positive* is the error probability that an immobile benign node is misidentified as a mobile malicious node.
- *False Negative* is the error probability that a mobile malicious node is misidentified as an immobile benign node.

We present the average results for 1000 runs of the simulation in each configuration. Each run is executed for 100 time slots. For each run, we obtain each metric as the average of the results of the SPRTs that are performed. Note that the SPRT will terminate if it decides that a node is a mobile malicious node.

First, we found that there were no observed false negatives for any value of ρ. As shown in Figure 5, the false positives rates were measured as at most 0.011. Thus, mobile malicious nodes were always detected and immobile benign nodes were misidentified as mobile malicious nodes with at most probability of 0.011. We point out that false positives mean that the misidentified nodes will be blocked only by the nodes that misidentify them, leaving other neighbors with which it can communicate.

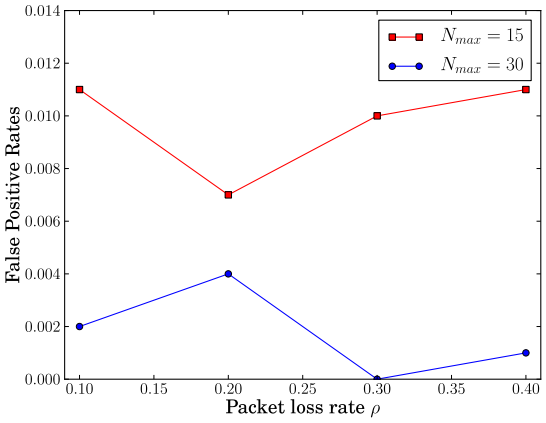

Figure 5: False positive rates vs. Packet loss rate (ρ) in the trueNegative case.

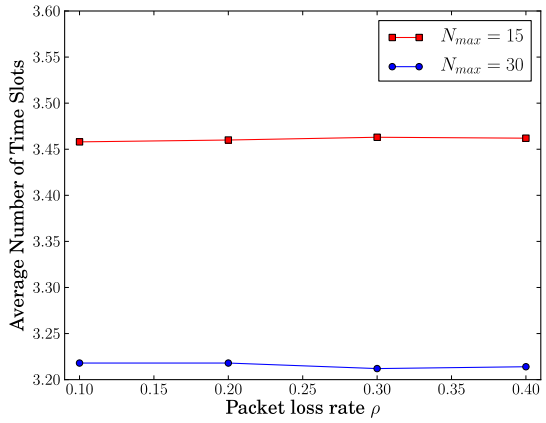

Figure 6: Average number of time slots vs. Packet loss rate (ρ) in the trueNegative case.

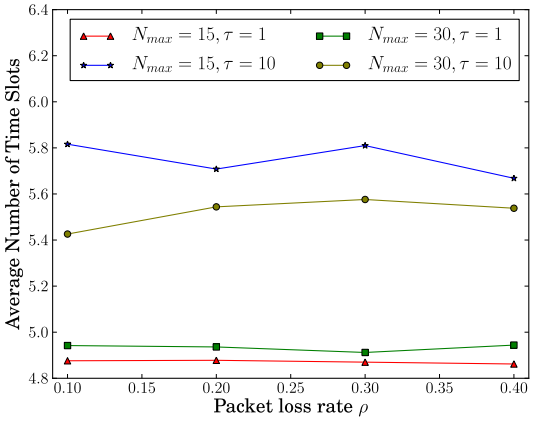

Figure 7: Average number of time slots vs. Packet loss rate (ρ) in the truePositive case with $\delta = 0$.

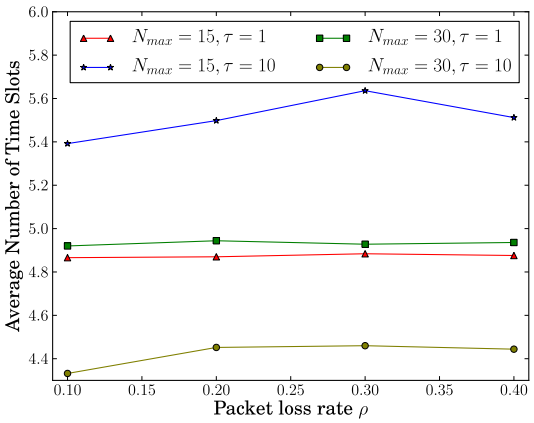

Figure 8: Average number of time slots vs. Packet loss rate (ρ) in the truePositive case with $\delta = 2$.

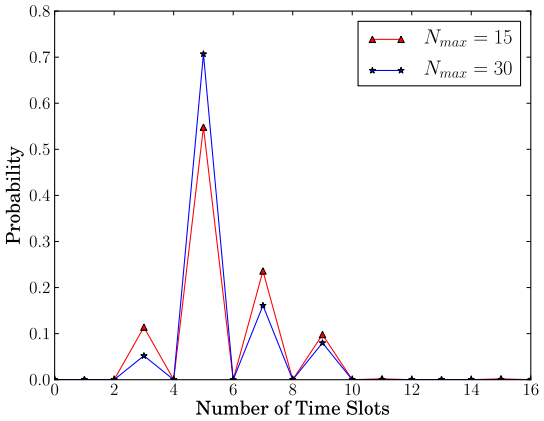

Figure 9: Probability distribution of number of time slots when $\rho = 0.4$ and $\tau = 10$ and $\delta = 0$.

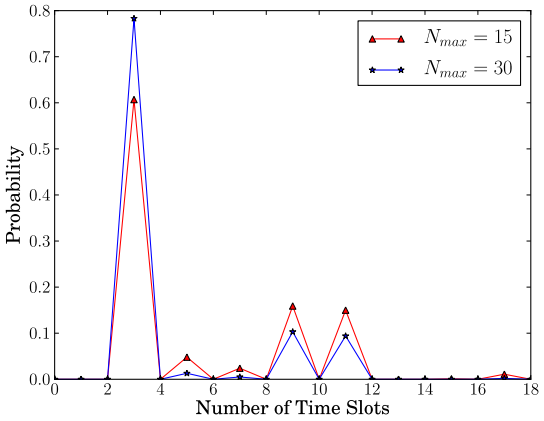

Figure 10: Probability distribution of number of time slots when $\rho = 0.4$ and $\tau = 10$ and $\delta = 2$.

Second, the average number of time slots to make a decision is shown in Figures 6, 7, and 8. We present the results for two cases. In the *trueNegative* case (Figure 6), node u decides correctly that an immobile benign node v is not a mobile malicious node. In the *truePositive* case (Figures 7 and 8), u correctly decides that mobile malicious node w has moved and should be blocked. Since we configure $\delta = 0, 2$, we consider two truePositive cases with $\delta = 0$ and $\delta = 2$.

In the trueNegative case, the average number of time slots reaches its maximum of 3.463 when $\rho = 0.3$ and $N_{max} = 15$. In case of the truePositive with $\delta = 0$, the average number of time slots reaches its maximum of 5.816 when $\rho = 0.1$, $N_{max} = 15$, and the mobile malicious node w stays close to node u for $\tau = 10$ time slots. In case of the truePositive with $\delta = 2$, the average number of time slots reaches its maximum of 5.636 when $\rho = 0.3$, $N_{max} = 15$, and the mobile malicious node w stays close to node u for $\tau = 10$ time slots. Thus, u reaches correct decisions in a few time slots in all cases. Moreover, we observe that the values of packet loss rate ρ do not substantially affect the average number of time slots in all cases. This indicates that an increase in ρ rarely affects the average number of time slots as long as not all packets are lost.

In the trueNegative case, the average number of time slots when $N_{max} = 30$ is less than when $N_{max} = 15$. This indicates that for smaller N_{max} in case of static benign node, the chance is greater that no packets are generated by v or w, leading to a rise in the average number of time slots for deciding that a benign node is not moving. In the truePositive case, the values of δ do not significantly affect the average number of time slots when $\tau = 1$. This means that the impact of δ on the decision process is very limited when mobile malicious node w stays within the vicinity of u for a short amount of time. On the other hand, when $\tau = 10$, an increase in δ results in a decrease in an average number of time slots for all values of N_{max} and ρ. From this observation, we infer that an increase in δ expedites the decision process to move toward H_1, leading to a decrease in the average number of time slots for mobile malicious node detection, even when w stays within the vicinity of u for a substantial amount of time.

Finally, Figure 9 (resp. Figure 10) shows the empirical distribution of the number of time slots in the truePositive case when $\rho = 0.4$, $\tau = 10$, and $\delta = 0$ (resp. $\delta = 2$). For this distribution, we examine two scenarios: $N_{max} = 15$ and $N_{max} = 30$. When $\delta = 0$, a total of 89.8% and 92% of the cases fall in the range from 3 to 7 time slots when $N_{max} = 15$ and $N_{max} = 30$, respectively. When $\delta = 2$, a total of 67.9% and 79.6% of the cases fall in the range from 3 to 7 and 3 to 5 time slots when $N_{max} = 15$ and $N_{max} = 30$, respectively. This implies that, in most cases, the number of time slots is less than or near the average and thus the SPRT detects mobile malicious nodes within at most seven time slots in most cases.

6. Conclusions

In this paper, we introduced the problem of mobile malicious nodes, which are a major threat to static sensor networks, even when immobile malicious nodes are detected and blocked. To address this threat in an effective and inexpensive way, we proposed a scheme for the distributed detection of mobile malicious node attacks. Our scheme applies the SPRT to detect nodes that have left a region and cannot send messages to their neighbors. We showed that our scheme detects any mobile malicious node quickly, and that a mobile malicious node can only avoid detection for a very limited time period. Finally, we evaluated the scheme through extensive simulation. The simulation results show that our scheme quickly detects mobile malicious nodes with just a few samples and with very low false positive and false negative rates.

References

[1] Y. Chen, W. Trappe, and R.P. Martin. Attack Detection in Wireless Localization. In *IEEE INFOCOM*, May 2007.

[2] X. Cheng, F. Liu, and D. Chen. Insider Attacker Detection in Wireless Sensor Networks. In *IEEE INFOCOM*, May 2007.

[3] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme. Robomote: enabling mobility in sensor networks In *IEEE IPSN*, April 2005.

[4] F. Delgosha and F. Fekri. Threshold key-establishment in distributed sensor networks using a multivariate scheme. In *IEEE INFOCOM*, April 2006.

[5] J. Deng, R. Han, and S. Mishra. Defending against Path-based DoS Attacks in Wireless Sensor Networks. In *ACM SASN*, November 2005.

[6] D. Dong, M. Li, Y. Liu, and X.Y. Li. Topological Detection on Wormholes in Wireless Ad Hoc and Sensor Networks. In *IEEE ICNP*, October 2009.

[7] A. Francillon and C. Castelluccia. Code Injection Attacks on Harvard-Architecture Devices. In *ACM CCS*, 2008.

[8] D.B. Faria and D.R. Cheriton. Radio Layer Security: Detecting identity-based attacks in wireless networks using signalprints. In *ACM WiSe*, September 2006.

[9] S. Ganeriwal and M. Srivastava. Reputation-based framework for high integrity sensor networks. In *ACM SASN*, October 2004.

[10] Q. Gu and R. Noorani. Towards self-propagate mal-packets in sensor networks. In *ACM WiSec*, 2008.

[11] C. Hartung, J. Balasalle, and R. Han. Node Compromise in Sensor Networks: The Need for Secure Systems. *Technical Report CU-CS-990-05, Department of Computer Science, University of Colorado at Boulder*, January 2005.

[12] Y.C. Hu, A. Perrig, and D.B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *IEEE INFOCOM*, April 2003.

[13] J. Jung, V. Paxon, A.W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *IEEE Symposium on Security and Privacy*, May 2004.

[14] J. Ho, M. Wright, and S.K. Das. Fast Detection of Replica Node Attacks in Sensor Networks Using Sequential Analysis. In *IEEE INFOCOM*, April 2009.

[15] Z.Li, W.Xu, R.Miller, and W.Trappe. Securing wireless systems via low layer enforcements. In *ACM WiSe*, September 2006.

[16] N. Patwari and S. K. Kasera. Robust location distinction using temporal link signatures. In *ACM MobiCom*, September 2007.

[17] J. Reich and E. Sklar. Robot Sensor Networks for Search and Rescue. In *IEEE International Workshop on Safety, Security and Rescue Robotics*, 2006.

[18] V. Shukla and D. Qiao. Distinguishing Data Transience from False Injection in Sensor Networks. In *IEEE SECON*, June 2007.

[19] M.Strasser, C. Popper, and S. Capkun. Efficient Uncoordinated FHSS Anti-jamming Communication. In *ACM MobiHoc*, May 2009.

[20] Y. Sun, Z. Han, W. Yu, and K. Liu. A trust evaluation framework in distributed networks: vulnerability analysis and defense against attacks. In *IEEE INFOCOM*, April 2006.

[21] D. Sy and L. Bao. CAPTRA: CoordinAted Packet TRAceback. In *IEEE IPSN*, April 2006.

[22] A. Wald. Sequential Analysis. *Dover Publications*, 2004.

[23] R. Wang, W. Du, and P. Ning. Containing Denial-of-Service Attacks in Broadcast Authentication in Sensor Networks. In *ACM MobiHoc*, September 2007.

[24] A.D. Wood, J.A. Stankovic, and S.H. Son. JAM: a jammed-area mapping service for sensor networks. In *IEEE RTSS*, December 2003.

[25] K. Xing, F. Liu, X. Cheng, and H.C. Du. Real-time Detection of Clone Attacks in Wireless Sensor Networks. In *IEEE ICDCS*, June 2008.

[26] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *ACM MobiHoc*, May 2005.

[27] F. Ye, H. Yang, and Z. Liu. Catching "moles" in sensor networks. In *IEEE ICDCS*, June 2007.

[28] J. Zhang, M. Firooz, N. Patwari, and S. K. Kasera. Advancing wireless link signatures for location distinction. In *ACM MobiCom*, September 2008.

[29] W. Zhang, M. Tran, S. Zhu, and G. Cao. A random perturbation-based scheme for pairwise key establishment in sensor networks. In *ACM MobiHoc*, September 2007.