

Sensor Node Compromise Detection: The Location Perspective*

Hui Song
Dept. of Computer Science
Frostburg State University
Frostburg, MD 21532
hsong@frostburg.edu

Sencun Zhu
Dept. of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
szhu@cse.psu.edu

Liang Xie
Dept. of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
lxie@cse.psu.edu

Guohong Cao
Dept. of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
gcao@cse.psu.edu

ABSTRACT

Node compromise is a serious security threat that hinders the successful deployment of large-scale wireless sensor networks. A node compromise often consists of three stages: physically obtaining and compromising the sensors, redeploying the compromised sensors, and compromised nodes launching attacks after their rejoining the network. By far, all the proposed compromise detection schemes address this problem at the third stage. In this paper, we make the first attempt to detect node compromise at the second stage. Our motivation is that for some applications an attacker may not be able to precisely deploy the compromised sensors back into their original positions. Thus, the detection of location change will become an indication of a potential node compromise. We name this *node redeployment detection* problem. We propose two approaches to detect node redeployment, based on the change of node neighborhood and the change of measured distances between nodes, respectively. Our simulation study shows that both schemes can detect node redeployment effectively (with low false positive rate and high detection rate).

Categories and Subject Descriptors

C.2 [Computer-Communication Networks]: Miscellaneous

General Terms

Security, Algorithm

Keywords

Wireless Sensor Networks, Node Compromise Detection

*This work was supported in part by Army Research Office (W911NF-05-1-0270) and the National Science Foundation (CNS-0524156, CNS-0519460, and CAREER-0643906).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IWCMC'07, August 12–16, 2007, Honolulu, Hawaii, USA.
Copyright 2007 ACM 978-1-59593-695/07/0009 ...\$5.00.

1. INTRODUCTION

Since sensors are cheap and have no tamper-proof hardware, an adversary may easily compromise them after capturing them [2]. Sensor node compromise is a serious security threat because a compromised node may launch attacks from different layers of the network protocol stack. For example, in the physical layer, a compromised node can jam the transmission channel. In the media access control layer, it can starve other nodes by manipulation of the random backoff mechanism. In the network layer, it can drop, alter, replay the routing messages. In the application layer, it can report false data to trigger false alarms [10], or attack the essential protocols for sensor networks (such as time synchronization, node localization [4], etc.). Thus, node compromise detection is a critical security requirement for the successful deployment of large-scale wireless sensor networks.

A node compromise attack often consists of three stages. The first stage is physically obtaining and compromising the sensors; the second stage is redeploying the compromised nodes back to the sensor network; and the last stage is compromised sensors rejoining the network and launching attacks. So for all the proposed compromise detection schemes [3, 8] address the node compromise problem at the third stage based on node misbehavior detection.

In this paper, we make three contributions. First, we make the first effort to address the node compromise problem at the second stage. Compromised nodes may be identified at the redeployment stage and prevented from rejoining the network. We refer to this as the *node redeployment detection* problem. Note that we are not aiming at completely solving the node compromise problem by our scheme alone. Instead, our goal is to prevent compromised nodes from easily rejoining the network. Even when compromised nodes have bypassed our defense, we still have the traditional node compromise detection schemes as the second line of defense. On the other hand, the detection result of our scheme may be used to trigger a verification process using the techniques such as software attestation [7].

Our second contribution is formalizing the node redeployment detection model, by exploiting the property of a stationary sensor network where the distance between two nodes does not change over time. The change of distance often indicates that the node is redeployed. Third, we propose two sets of solutions to detect node redeployment: a neighborhood-based approach and a distance-based approach, both of which work in a localized and distributed fashion. In the first approach, the redeployment detection is based on the change of neighborhood (in this paper we use transmission

power levels as an indicator). In the second approach, the redeployment detection is based on the change of distances. The technique of unpaired observations is used to detect the distance change. Our simulation results indicate that our proposed schemes can effectively detect node redeployment.

The rest of the paper is organized as follows. In Section 2, the assumptions and the formalized node redeployment detection model are presented. Section 3 and 4 present the two proposed approaches, respectively. The performance of these two approaches are evaluated in Section 5 and Section 6 concludes the paper.

2. SYSTEM AND ATTACK MODEL

Security Assumptions We assume that each sensor node is assigned a unique id before deployment and it can authenticate the messages sent or received with appropriate shared keys established through a key management protocol [9]. More specifically, to secure the exchanged messages, the message sent between sensors should be protected by the pairwise key shared between them; and the local broadcast packets can be protected by a cluster key as in the LEAP scheme [9].

Attack Model An attacker can obtain the sensors physically in several ways. For example, the attacker can capture the sensors from the sensing field in person. They can also collect sensors from the deployed area using unmanned vehicles. For the purpose of placing the sensors back to the original locations after compromising, the attackers might record the sensors' original locations. After obtaining the sensors, the attackers tamper the sensors and obtain the critical information (such as the cryptographic keys, the collected sensitive data, etc.) in the sensors. The attacker can even add malicious code—which can be used to attack the sensor network after rejoining the network—into the sensors. The adversary then redeployes the compromised nodes back into the network. To minimize the chance of being detected, the attacker tries to put the compromised sensors back to the original locations based on the recorded location information before. In the following, we refer to this activity as *node redeployment attack* (or simply *node redeployment*).

An attack similar to what we are addressing here is called *node clone attack*, in which the replicas of compromised sensors are inserted at strategic locations to launch attacks. This attack can be addressed effectively by discovering the existence of duplicate node ids in the network [1]. However, the redeployment attack cannot be detected by the countermeasures for clone attack, since the compromised, original sensors are used for attacking.

Design Rationale The recorded location information (e.g., based on GPS devices) can assist the attacker to redeploy the compromised sensors back to the original locations. Currently, two GPS positioning services of different precision were available. For the more-accurate one, called Precise Positioning Service (PPS, which was available only for the U.S. army), the position estimation error was about $\pm 18 m$ horizontally and $\pm 23 m$ vertically [5]. According to the U.S. Department of Defense, the less-precise service, called Standard Positioning Service (SPS, which was available for civil use), has a horizontal accuracy of $\pm 100 m$ at 99.5% confidence interval and $\pm 300 m$ at 99.9% [5]. Thus, because of the measurement errors of the GPS system and the operational errors when redeploying the sensors (either by hand or by unmanned vehicles), it is often impossible to redeploy the compromised sensor back to the exact, original location.

Note that the attackers often do not have full control of the environment where the sensors are deployed. This implies that, even if the attacker can mark the positions when obtaining the sensors, they may not be able to put the compromised sensors back to their original positions exactly. Battle field is such an example scenario.

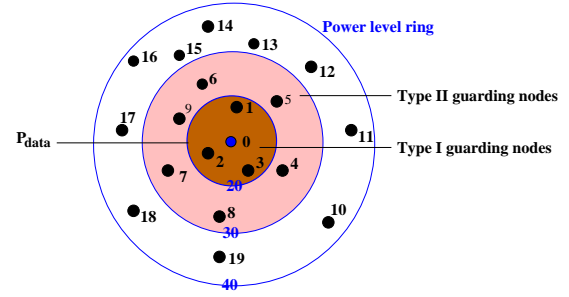


Figure 1: Setting up power-level tuple at the neighbors of node 0. In the figure, the rings represent node 0's coverage using transmission power levels 20, 30, and 40, respectively. We can see that nodes 1–3 have a tuple of $\langle \{0\}, 10 \rangle$, nodes 4–9 have a tuple of $\langle \{0\}, 20 \rangle$, and nodes 10–19 have a tuple of $\langle \{0\}, 30 \rangle$.

We assume that the attacker can only depend on techniques such as GPS positioning when redeploying the sensors. Our detection model and solutions are targeted for such scenarios.

Next, we present two redeployment detection schemes: one is a neighborhood-based scheme and the other is a distance-based scheme.

3. NEIGHBORSHIP-BASED DETECTION

In this scheme, each node has a set of monitoring nodes. For ease of presentation, we refer to the node being monitored as *monitee* and the nodes that monitor it as *monitors* in the rest of the paper.

The basis of the neighborhood-based scheme is that different transmission power levels correspond to different transmission ranges. Thus, a certain neighbor can hear from the monitee only when the monitee transmits packets with a transmission power higher than a certain level. By having the monitee transmitting with all possible transmission power levels, each of its neighbors can record the power level from which the packets sent from the monitee start to be heard (note that any message sent using a power level higher than that can also be heard). If later a neighbor can overhear a packet from the monitee with a lower power level or only with a higher power level, it can suspect that the monitee is redeployed.

3.1 Scheme Description

This scheme consists of three phases: power-level tuple recording, guardian node marking, and redeployment detection. The first two phases happen shortly after the initial deployment of the sensor network (when there is no attackers) and is executed only once, whereas the third phase can be executed periodically or for multiple times, depending on the detection requirements. In the following, we present these three phases in detail one by one.

Power-level Tuple Recording When sensors are first deployed in the field, A sensor k sets up power-level tuple at its neighboring nodes as follows. After booting up, k broadcasts a special packet, called *probe*, using all possible power levels: from the minimum to the maximum (e.g., from 1 to 255 for Mica2 motes). The probe contains three fields: k (the node id), p_i (the current power level that is used for transmission), and p_{data} (the power level that will be used for data communication after joining the network). Note that, p_{data} is fixed for a node, although it could be different for different nodes. For ease of presentation, we assume that all the nodes have the same p_{data} . As we will see the next subsection, p_{data} is used for guardian node marking and can be dynamic as well.

when a neighbor of k hears a probe from k for the first time, that node retrieves the power-level information from the probe and records it as a node-power pair $\langle k, p_{min} \rangle$. Since a node could be

the monitoring node for multiple nodes, it can combine the node-power pairs into power-level tuples based on p_{min} to save storage space. In this way, eventually, each node will get a set of power-level tuples, $\langle \{nodelist\}, p_{min} \rangle$.

Further, to decrease the number of probing rounds, node k can increase the power level by p instead of by one each time. Suppose the maximum transmission power level is p_{MAX} and $p_{MAX} = n * p$, then each node only needs to send the probe for n rounds. For example, if we increase the power level by 16 each time, then, for Mica2 motes, only 16 rounds of probing are needed for a sensor.

Fig. 1 illustrates this procedure by a simple example, where $p = 10$ and the power-level tuples for node 0 is recorded at its neighboring nodes.

Guardian Node Marking In this phase, given p_{data} , the *guardian nodes* for a monitee are self-discovered and marked based on the recorded p_{min} tuples. For a monitee, its guardian nodes are those nodes whose p_{min} is g -level around p_{data} —that is, $|p_{min} - p_{data}| \leq g$. For example, in Fig. 1, node 0's guardian nodes are those nodes that are within the shadowed area with $g = 1$ when $p_{data} = 20$. Note that, when g is big enough, all the neighbors of a monitee become guardian nodes. We can tune the value of g to achieve a balance between performance and overhead. In this work, we set the required number of guardian nodes to six (based on our simulations).

Note that, in our scheme, only the guardian nodes (for a monitee) need to monitor the monitee actively. The reasons are twofold. First, since a monitee sends its regular packets using power level p_{data} , it is the guardian nodes who are capable of and more sensitive on detecting the redeployment than the other neighbors. Second, since only part of the neighbors are monitoring the monitee, the monitoring overhead is reduced.

The guardian nodes are further classified into two types: *Type I* (those with $max(p_{data} - gp, p) \leq p_{min} \leq p_{data}$) and *Type II* (those with $p_{data} < p_{min} \leq min(p_{data} + gp, p_{max})$). For example, in Fig. 1, the Type I guardian nodes are those within heavily-shadowed area, whereas Type II are those within the lightly-shadowed area. The purpose of classifying guardian nodes is explained in the next phase. Note that if p_{data} is changed dynamically (e.g., due to power control purpose), the monitee needs to broadcast a p_{data} change notification packet (which contains the new p_{data}) to all of its neighbors using the transmission power level p_{max} . In this way, the guardian nodes can be reselected and remarked adaptively.

By far, each node gets a set of power-level tuples for the nodes that it is monitoring; in addition, the guardian nodes for each node are selected and marked. Next, we describe how to use the collected power-level tuples and the guardian nodes in redeployment detection.

Redeployment Detection The redeployment of a monitee is detected by its guardian nodes. The guardian nodes keep on monitoring the packets sent from the monitee. Based on its guardian type, guardian nodes detect redeployment in different ways. More specifically, Type I guardian nodes detect the redeployment of a monitee when they can no longer hear from the monitee, whereas Type II guardian nodes detect the redeployment when they receive a message from the monitee with a transmission power level smaller than the recorded p_{min} for it. For instance, in Fig. 2, node 6 is a Type II guardian node. Its power-level tuple for node 0 is $\langle \{0\}, 30 \rangle$. Suppose node 0 is redeployed into position 0'. Now node 6 can hear node 0 when node 0 uses transmission power level 20, which is less than the previously recorded p_{min} (i.e., 30). Thus, node 6 can suspect that node 0 is compromised and redeployed. However, node 3 (a Type I guardian node) suspects that node 0 is redeployed because it can no longer hear from node 0 with power

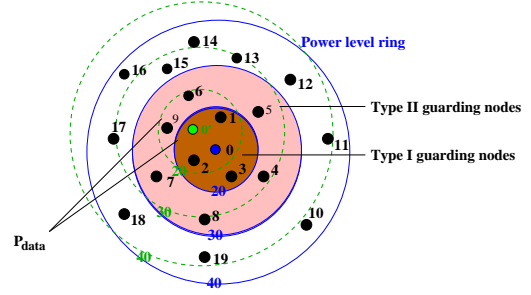


Figure 2: Redeployment detection for node 0. In the figure, suppose node 0 is redeployed at position 0' and the dashed rings shows the new coverage area with different transmission power levels. When p_{data} is 20, several nodes can detect the redeployment of node 0, including Type I node 3 and Type II nodes 6 and 9, whereas it is nodes 8 and 15 when p_{data} is 30 and node 10 when p_{data} is 40.

level p_{min} , which is 20. Note that only part of the guardian nodes can detect the redeployment of the suspicious node.

A redeployed compromised node can adjust the real transmit power level, either higher or lower than the claimed p_{data} , in attempt to eluding our detection. Our scheme can still detect node redeployment effectively (see Subsection 3.2 for more discussions on this issue).

A single node, however, could make false detections, because the redeployment detection is based on RF (or radio) signal, which is subject to the surrounding environments and has errors. We also notice that, it is simple to use a cluster head to collect the detection decisions from all monitors and make a final decision on the redeployment of the monitee. However, if the cluster-head is compromised, the whole redeployment detection fails. Thus, we cannot depend on one node to make the final decision. Instead, the final decision making process should be a distributed one.

To achieve these goals, each node that detects a redeployment broadcasts an announcement packet to notify other nodes of its detection, using the maximum transmission power level. Each monitor counts how many distinct detection announcements it has received during a fixed period of time. If a monitor receives three or more detection announcements from distinct nodes, that monitor node treats the accused node as redeployed. We use three nodes as the detection threshold because we notice that, with three or more monitoring nodes, a compromised node must be put back to the original location to elude detection. Thus, here we simply use three as the detection threshold.

Note that, the redeployment detection is a distributed procedure—i.e., the detection decision is made at each node individually. The distributed property avoids the security weakness of depending on a single node (e.g., a cluster head) to make the final decision for all nodes. In addition, when redeployment is detected, we can use other techniques such as software attestation [7] to further verify the node compromise, or exclude the suspicious node from future communication.

3.2 Security Analysis

A compromised (or redeployed) monitee may attack our scheme by adjusting the real transmission power level. That is, the redeployed monitee uses a power level other than p_{data} —either higher or lower—to send regular packets. Our proposed scheme, however, can tolerate such an attack strategy and detect the redeployment effectively. On one hand, if the redeployed node is using a transmission power level smaller than p_{data} , this attack can be detected by certain Type I guardian nodes because they cannot receive

any messages at all. On the other hand, if the real transmission power level is higher than p_{data} , more Type II guardian nodes will hear the messages and detect the redeployment. Thus, adjusting transmission power level cannot elude our neighborhood-based redeployment detection (except for some edge nodes who may not have enough monitors).

When multiple nodes in a close neighborhood are compromised and redeployed back together, they may collude to launch two attacks. In the first attack, the compromised nodes collude to avoid redeployment detection. However, as long as three or more nodes confirm the detection (of each of them), their redeployment can still be detected. Thus, our scheme is resilient to such an attack. The second attack is that multiple compromised nodes collude to accuse a good node to be redeployed. Since our scheme has high detection rate (see Section 5), we assume that each node has less than three rejoined compromised nodes (that has eluded the redeployment detection) around its neighborhood. Our distance-based scheme presented in the next section is more resilient to this falsely-accusing attacks and does not make this assumption.

4. DISTANCE-BASED DETECTION

In the previous scheme, the node redeployment detection is based on the change of transmission power level recorded at the monitee's neighbors (or monitors). In this scheme, redeployment detection is based on the change of distances measured at the monitors. Note that different localization techniques, including time of arrival, time difference of arrival, angle of arrival, and received signal strength (RSSI), can be used here to measure the distance, and our scheme does not rely on a specific one.

The distance-based redeployment detection is based on the following insight: without node redeployment attack, the monitee's location will not change and the distance measurements at its neighboring nodes should be consistent, respectively. Thus, if the monitee is redeployed into a different location, a monitor can detect the redeployment by noticing the inconsistency in the distance measurements. In other words, the distributions of the distance measurements before and after redeployment are different.

More specifically, if a monitor makes *before* and *after* distance measurements on the monitee, it can calculate the difference between the *before* and *after* distance-measurement pairs and determine whether that monitee has been redeployed or not. The *before* measurement, which consists of a set of distance measurements, is collected right after the initial deployment, and we denote the set of data in *before* measurement as the *reference-set*. The *after* measurement is done at sometime thereafter and we denote the collected data set at this time as the *testing-set*.

Several statistical techniques can be used to test/compare the distributions of two sets of data. In this work, we adopt the *unpaired observations* technique [6] to detect redeployment, which has been widely used in testing a hypothesis based on the difference between sample means. Next, we present our proposed scheme in detail.

4.1 Scheme Description

In this scheme, a monitor first collects the reference-sets for its monitees, one for each. To test whether a monitee is redeployed, the monitor collects a testing-set and compares it with the reference-set. If the two sets of data fail the test by unpaired observations, we claim that a node redeployment is detected.

Specifically, our scheme has three phases: a *training phase*, a *detecting phase*, and a *verification phase*. The training phase happens shortly after the initial deployment of the sensor network (when there are no attackers) and is executed only once, whereas the detection and verification phases can be executed periodically or for multiple times, depending on the detection requirements. The train-

ing phase collects the reference-set, which corresponds to the *before* measurement in the unpaired observations; the detection phase collects the testing-set(s), which corresponds to the *after* measurement; the verification phase makes a distributed voting among the monitors of a monitee. Next, we describe the three phases in detail. **The Training Phase** In this phase, a monitee first discovers its monitors by broadcasting a *join* packet to its neighbors. A neighboring node who receives the join packet should send back a *join-reply* packet to the monitee. The monitee waits for a period of time until it receives enough number of join replies. When the monitee receives enough join replies, the monitor discovery process terminates. Otherwise, to discover more monitors, the monitee increases its transmission power level and sends another join message. This process is repeated until it either finds enough number of monitors or reaches the maximum transmission power level. Then the monitee broadcasts N beacon messages to its monitors. Each monitor measures and records the measured distances of the N messages as $\{D_{i1}, D_{i2}, \dots, D_{iN}\}$, which is the *reference-set*.

The Detection Phase Periodically, a monitor (node i) collects a series of n distance measurements $\{d_{i1}, d_{i2}, \dots, d_{in}\}$ for each of its monitees, respectively. Each set of data is a testing-set. Then the unpaired observations technique is used to test the two sets of data (the reference-set and the testing-set) for redeployment detection. If they fail the test, the monitor claims that the monitee is redeployed, since the difference between the two sets of data is significant. Further, to reduce the false positive rates due to environmental dynamics and measurement errors, the monitor can collect multiple testing-sets and run the test several times before claiming a redeployment detection.

The Verification Phase To further decrease the false alarm rate, we follow the detection phase with a verifying phase. When a monitor detects a redeployment, it broadcasts its detection decision (or called *detection report*) to its neighbors using the maximum transmission power level. Upon hearing of a detection report, each monitor (for the same monitee) broadcasts its detection report too. Each monitor collects the detection reports from other monitors. If a monitor collects a threshold number of positive detection reports from others, it will claim that the monitee is redeployed. Then the monitor can take further actions to isolate the monitee from its future communications. Note that, the redeployment verification is done distributively.

Another way to do verification is to run the detection several times. If only a small percentage of them do not confirm the redeployment, the redeployment is verified. In our simulation, we adopt this approach to evaluate the performance of this scheme.

4.2 Security Analysis

Similar to the previous scheme, multiple malicious, compromised monitors may collude together to launch two kinds of attacks. Like the previous scheme, our distance-based scheme is resilient to the first attack (i.e., eluding detection). Regarding the second attack (i.e., falsely-accusing attack), since the redeployed compromised nodes can be detected more effectively (see Section 5), this scheme is more resilient to this attack than the previous scheme.

5. PERFORMANCE EVALUATIONS

5.1 Simulation Setup

We evaluate the performance of the two redeployment detection schemes by simulation. TOSSIM is a discrete event simulator for TinyOS sensor networks, where one can debug, test, and analyze algorithms in a controlled and repeatable environment. Tython provides a scripting interface to TOSSIM, which has greatly enhanced TOSSIM's functionality of simulating wireless sensor networks. In

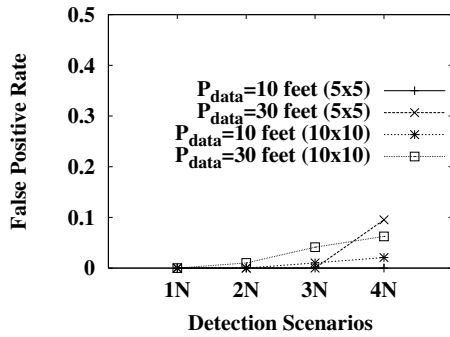


Figure 3: The FP rate of the neighborhood-based scheme for two topologies with different node redeployment scenarios.

the simulation, we utilize the tools (such as LossBuilder, DiscSensorModel, EmpiricalModel, etc.) provided by Tython to generate the deployment topology and the lossy model for simulating the sensor network.

In the simulation, we consider the deployment area as a 100 feet by 100 feet square plane. Since our schemes are localized schemes, a large deployment topology is not necessary: The redeployment detection is done within a local neighborhood of nodes. Using Tython, we generate two (grid+random) topologies, one with 25 nodes and one with 100 nodes. The former represents a sparse sensor network which we refer to as *5x5 topology*; the latter represents a dense sensor network and is referred to as *10x10 topology*. In the simulations, we assume that the monitors use the received signal strengths (RSSIs) to represent the distance measurements for the monitee, although other distance measurement techniques can be used here as well.

The number of compromised nodes (randomly picked) varies from one to four in a network of 25 or 100 nodes. Again, because of the distributed and localized properties of our schemes, the performance results showed below can be zoomed out to capture the performance in larger networks with more compromised nodes.

For each of the two topologies, we use four redeployment scenarios to evaluate the performance of our proposed schemes. In the following figures, we use “ xNy ” to represent the y -th scenario with x node(s) being redeployed. When y is omitted, it shows the average result for several rounds of testing. For instance, “1N3” means the third scenario with one node redeployed and “2N” means the average results of several runs with two nodes being redeployed. In the simulation, node 1 to node 4 are the possible compromised nodes and Table 1 shows the redeployed distances of them in different scenarios in the two topologies.

Table 1: The Redeploy distances (in feet) in different scenarios.

	5x5 Topology				10x10 Topology			
	1N	2N	3N	4N	1N	2N	3N	4N
Node 1	2.0	2.0	2.0	2.0	4.1	4.1	4.1	4.1
Node 2		2.5	2.5	2.5		3.5	3.5	3.5
Node 3			4.6	4.6			4.7	4.7
Node 4				3.9				3.5

Two metrics are used to evaluate the performance: the detection rate and the false positive rate. The former tells the percentage of node redeployment that can be successfully detected by our system, whereas the latter shows the percentage of false detections.

5.2 Results for the Neighborhood-based Scheme

To evaluate the neighborhood-based scheme, we suppose that each node can use four transmission power levels—the corresponding

transmission ranges are 10, 20, 30, and 40 feet, respectively. All the nodes in the network use the same power level, p_{data} , when transmitting regular packets. For a node, all of its neighbors function as the guardian nodes for it.

The False Positive Rate Fig. 3(1) shows the false positive rate of the neighborhood-based scheme. From the figure, we can see that, in both of the topologies, the FP rates are lower than 10%. In addition, the smaller the p_{data} is, the lower the FP rate. This can be explained as follows. In this work, we are considering the case that the attackers try to redeploy the sensors back to the original locations to elude detection. Since the location displacement is relatively small, the nodes close to the redeployed node can detect the redeployment more effectively. Thus, the smaller p_{data} is, the closer the guardian nodes to the redeployed node and the more accurate the redeployment detection is. This explains why the false positive rate is lower for smaller p_{data} .

Table 2: The detection rate of the neighborhood-based scheme for two topologies with different node redeployment scenarios.

For the 5x5 topology:

Scenarios	P_{data} (feet)	Detection Rate
1N	10, 30	0 (for all)
2N	10, 30	100% (for all)
3N	10, 30	100% (for all)
4N	10, 30	50%, 50%

For the 10x10 topology:

Scenarios	P_{data} (feet)	Detection Rate
all	10, 30	100% (for all)

The Detection Rate Table 2 shows the detection rate of the neighborhood-based scheme. It can be seen that, the detection rate is not satisfiable for the 5x5 topology in some of the scenarios, whereas 100 percent detection rate is achieved for the 10x10 topology. The reason is that, in the neighborhood-based scheme, the redeployment detection depends on the change of P_{min} observed at the neighbors of a monitee. If the monitee does not have sufficient neighboring nodes, the effectiveness of the neighborhood-based scheme will be low. Fig. 4 shows the histogram of the number of neighbors (at each node) in the two topologies, where we can get more hints on explaining Table 2.

Fig. 4(1) shows the neighbor number histogram in the 5x5 topology (with $P_{data} = 30$ feet). From the figure, we can see that many nodes only have a few neighbors (e.g., one node only has three neighbors). If such a node is redeployed, its redeployment cannot be detected effectively because of lacking enough number of neighbors. The scenario with one node redeployed is such a case, where the redeployed node only has five neighbors, resulting a detection rate of zero. However, our scheme is highly effective in dense sensor networks such as in the 10x10 topology. From Fig. 4(2), we can see that, in the 10x10 topology, each node has more than five neighbors, resulting a high detection rate.

The results above implies that the neighborhood-based scheme is more effective in a dense sensor network where a node can have abundant number of neighbors at different transmission ranges. Next, we present the performance results for the distance-based scheme.

5.3 Results for the Distance-based Scheme

The False Positive Rate Fig. 5 shows the performance of the distance-based scheme when one, two, three, or four nodes are redeployed in a 5x5 deployment topology. The corresponding redeployment distances are shown in Table 3. Recall that “ xNy ” represents the y -th scenario with x node(s) being redeployed; while “ xN ” shows the average result for several rounds of testing with x node(s) being redeployed. From Fig. 5(1), we can make the following observa-

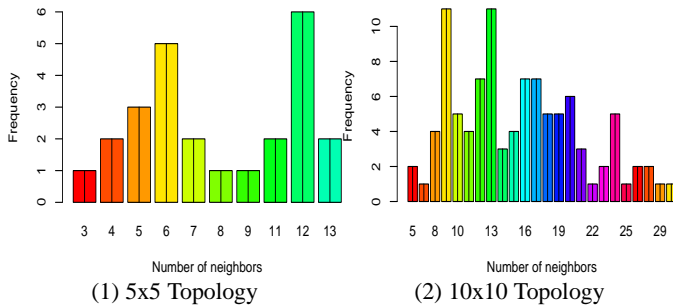


Figure 4: The histograms for the number of neighbors (at each node) in the two topologies.

tions. First, the distance-based scheme can achieve a low FP rate (less than 4% most of the time). Second, the smaller thr is, the lower the FP rate is. Here thr represents the threshold percentage of negative detection (i.e., no detection) out of k runs (i.e., $\frac{k-thr}{k}$, and $k = 10$ in the simulation) at one monitoring node. That is, only when the percentage of negative detection is smaller than thr will the node raise a detection alarm. This explains why a smaller thr results in a lower FP rate.

The Detection Rate Fig 5(2) shows the detection rates for the proposed scheme in a 5x5 topology. From the figure, we can see that, in most of the redeployment scenarios, our scheme achieves 100 percent detection even when thr is very small (e.g., 0.05 or 0.1). However, when the redeployment distance is too small (e.g., scenario 2N), some redeployments may not be detected. Generally speaking, the results show that the distribution change of the distance measurements can reflect the location change (or redeployment) precisely. The high detection rate also indicates that the distance-based approach works well even in sparse sensor networks. Note that, in the simulation, we did not consider message loss, thus the detection rate will be lower than 100 percent in real environments.

Note that we have also evaluated the performance of the distance-based scheme in a dense sensor network such as the 10x10 topology. The results show that we can always achieve 100 percent detection range and 0 percent false positive rate in all of eight redeployment scenarios. These figures are not included in the paper. The above results suggest that the distance-based scheme works well in both sparse and dense sensor networks.

Table 3: The Redeploy distances (in feet) in different scenarios.

	5x5 Topology						
	1N1	1N2	1N3	1N4	2N	3N	4N
Node 1	4.8	7.7	10.3	13.7	2.0	2.0	2.0
Node 2					2.5	2.5	2.5
Node 3						4.6	4.6
Node 4							3.9

As a final comment, the distance-based scheme achieves a better performance than the neighborhood-based scheme at the cost of higher communication overhead. More specifically, in the distance-based scheme, all the neighbors are involved in the monitoring and detecting process, whereas in the neighborhood-based scheme, only part of the neighbors (i.e., the guardian nodes) are involved. Thus, there is a tradeoff between performance and overhead and the two schemes can be adopted accordingly.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose to detect node compromise attacks right after the redeployment of the compromised nodes, whereas in the previous work detection happens after the compromised nodes

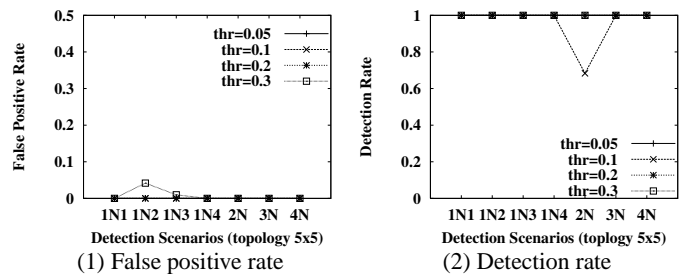


Figure 5: The performance of the distance-based scheme (in terms of false positive rate and detection rate) for a 5x5 topology with different node redeployment scenarios.

have rejoined the network. By detecting and isolating the compromised nodes early, we can prevent the compromise nodes from launching malicious attacks from different layers of the protocol stack. On the other hand, our scheme can be integrated with the previously proposed schemes to defend against node compromise attacks. We proposed two approaches to detect the redeployment attack. Our performance results show that both of the proposed schemes have high detection rate and low false positive rate. In addition, we notice that our neighborhood-based scheme performs better in a dense sensor network than in a sparse one, while the distance-based scheme works well in both dense and sparse sensor networks. As our future work, we will implement the proposed schemes in real sensors (e.g., MICA2 motes) and conduct experiments to evaluate their performance. We will also discuss the effect of other factors (such as irregular transmission range, battery level, etc.) on our schemes and evaluate them using additional performance metrics such as storage overhead, communication overhead, computation overhead, etc.

7. REFERENCES

- [1] H. Choi, S. Zhu, and T. Laporta, "SET: Detecting Node Clones in Sensor Networks," in *SecureComm*, Sep. 2007.
- [2] J. Deng, R. Han, and S. Mishra, "Countermeasures against traffic analysis attacks in wireless sensor networks," *IEEE SecureComm*, pp. 113–126, September 2005.
- [3] P. Kyasanur and N. H. Vaidya, "Detection and handling of mac layer misbehavior in wireless networks," in *IEEE DSN*, 2003.
- [4] D. Liu, P. Ning, and W. Du, "Attack-resistant location estimation in sensor networks," in *ACM IPSN '05*, 2005.
- [5] A. Michalski and J. Czajewski, "The accuracy of the global positioning systems," *IEEE Instrumentation & Measurement Magazine*, vol. 7, no. 1, pp. 56–60, March 2004.
- [6] NIST/SEMATECH, "e-handbook of statistical methods," <http://www.itl.nist.gov/div898/handbook/toolaids/pff/prc.pdf>.
- [7] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "Swatt: Software-based attestation for embedded devices," in *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.
- [8] F. Ye, H. Yang, and Z. Liu, "Catching Moles in Sensor Networks," in *IEEE ICDCS*, Jun. 2007.
- [9] S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *ACM CCS*, 2003.
- [10] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in *Proceedings of IEEE Symposium on Security and Privacy*, 2004, pp. 259–271.