

A FRAMEWORK FOR ROBUST DETECTION AND PREVENTION  
OF WIDE-SPREAD NODE COMPROMISE IN  
WIRELESS SENSOR NETWORKS

by  
JUN-WON HO

Presented to the Faculty of the Graduate School of  
The University of Texas at Arlington in Partial Fulfillment  
of the Requirements  
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2010

UMI Number: 3408924

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3408924

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

Copyright © by JUN-WON HO 2010

All Rights Reserved

## ACKNOWLEDGEMENTS

I would like to thank my supervising professors Sajal K.Das and Matthew Wright for their invaluable advices and generous financial supports during the course of my doctoral studies.

I would also like to appreciate to the Korean Government for providing financial support during my doctoral studies. I would not have completed a Ph.D. degree without the IT Scholarship for Graduate Study in Overseas from the Korean Government.

Lastly but not leastly, I would like to express my deep gratitude to my parents, Mr. Jung-Sik Ho and Mrs. Jung-Ja Park, and my elder brother, Dr. Sung-Jin Ho who have inspired and encouraged me to start Ph.D. study, and helped me overcome difficult times whenever I experienced them in the course of my doctoral studies. I would not have even dreamed to start and complete Ph.D. study without the support of my family.

APRIL 16, 2010

## ABSTRACT

# A FRAMEWORK FOR ROBUST DETECTION AND PREVENTION OF WIDE-SPREAD NODE COMPROMISE IN WIRELESS SENSOR NETWORKS

JUN-WON HO, Ph.D.

The University of Texas at Arlington, 2010

Supervising Professors: Sajal K. Das, Matthew Wright

Wireless sensor networks are known to be vulnerable to a variety of attacks that could undermine normal sensor network operations. Many schemes have been developed to defend the wireless sensor networks against various attacks. Most of them focus on making the network and service protocols be attack-resilient rather than rooting out the source of attacks. Although the attack-resiliency approach mitigates the threats on sensor network protocols, it requires substantial time and effort for continuously enhancing the robustness of the protocols in line with the emergence of new types of attacks. Accordingly, if we are able to detect and remove the sources of attacks as soon as possible, we could save the large amount of time and effort incurred from employing the attack-resiliency approach. In wireless sensor networks, the principle sources of various attacks are compromised nodes. Specifically, since sensor nodes are deployed in an unattended manner, an adversary can physically capture and compromise sensor nodes, and mount a variety of attacks with the compromised nodes. He can also move the compromised nodes to multiple locations to

evade the detection. Moreover, he can create wide-spread influence by generating many replica nodes of a few compromised nodes or propagating malicious worm into the network. Our works are designed for rooting out the sources of possible threats by quickly detecting and removing compromised nodes and preventing wide-spread node compromise through replica node and worm propagation attacks.

To meet this challenge, we propose a framework for robust detection and revocation of wide-spread node compromise in wireless sensor networks. In the framework, we first propose a reputation-based trust management scheme to facilitate static node compromise detection, and then propose a distributed detection scheme to achieve fast mobile node compromise detection, and finally propose replica node detection and worm propagation detection schemes to prevent wide-spread node compromise. Specifically, the framework is composed of five components. In the first component, we quickly detect the suspected regions in which compromised nodes are likely placed and perform software attestation against the nodes in the suspected regions, leading to the detection and revocation of the compromised nodes. However, if the attacker moves the compromised nodes to multiple locations in the network, such as by employing simple robotic platforms or moving the nodes by hand, he can evade the detection scheme in the first component. To resolve this limitation, we propose the second component in which we quickly detect these mobile malicious nodes that are silent for unusually many time periods—such nodes are likely to be moving—and block them from communicating in fully distributed manner.

To reduce the time and effort incurred from directly compromising many benign nodes, attacker may launch replica node attacks in which he generates many replica nodes of a few compromised nodes and widely spread them over the network. To thwart wide-spread node compromise by replica node attacks, we propose two complementary schemes for replica detection as the third and fourth components. In

the third component, we detect static replica nodes by leveraging the intuition that static replica nodes are placed in more than one location. In the fourth component, we quickly detect mobile replicas by leveraging the intuition that mobile replicas are in two or more locations at once and thus appear to move much faster than benign nodes, leading to highly likely exceed the predefined maximum speed.

However, the attacker needs to prepare as many sensor nodes as the number of replicas that he wants to generate in replica node attacks. Thus, the attack costs will increase in proportion to the number of deployed replicas. To reduce these costs, the attacker may attempt to widely spread node compromise by capturing a few nodes and having the captured nodes propagate malicious worm through the network, leading to the fast compromise of many benign nodes. To fight against this type of attack, we propose the fifth component in which we quickly detect worm propagation in fully distributed fashion by leveraging the intuition that a worm's communication pattern is different from benign traffic.

Through analysis and experimental study, we show that these components achieve robust and effective detection and revocation capability of node compromise, replica node, worm propagation with reasonable overhead.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . .	iii
ABSTRACT . . . . .	iv
LIST OF FIGURES . . . . .	xii
LIST OF TABLES . . . . .	xv
Chapter	Page
1. INTRODUCTION . . . . .	1
1.1 Problem Statement . . . . .	1
1.2 Challenges . . . . .	4
1.3 Contributions . . . . .	5
1.4 Organization . . . . .	7
2. BACKGROUND . . . . .	9
2.1 Group Deployment Strategy . . . . .	9
2.2 Sequential Probability Ratio Test . . . . .	10
2.3 Random Mobility Model . . . . .	13
2.4 Related Work . . . . .	13
3. ZONETRUST: ZONE-BASED NODE COMPROMISE DETECTION . . .	19
3.1 Overview . . . . .	19
3.2 Assumptions . . . . .	21
3.3 Protocol Description . . . . .	22
3.3.1 Phase I: Zone Discovery and Trust Aggregator Selection . . .	24
3.3.2 Phase II: Trust Formation and Forwarding . . . . .	24
3.3.3 Phase III: Detection and Revocation . . . . .	26



3.4	Security Analysis . . . . .	28
3.4.1	Detection Accuracy . . . . .	28
3.4.2	Limitations of Node Compromise Attacks . . . . .	29
3.5	Performance Analysis . . . . .	34
3.5.1	Communication Overhead . . . . .	34
3.5.2	Computation and Storage Overhead . . . . .	35
3.6	Experimental Study . . . . .	36
3.6.1	Simulation Environment . . . . .	36
3.6.2	Simulation Results . . . . .	37
3.7	Summary . . . . .	40
4.	MOBILE MALICIOUS NODE DETECTION . . . . .	41
4.1	Overview . . . . .	41
4.2	Assumptions . . . . .	44
4.3	Protocol Description . . . . .	46
4.4	Security Analysis . . . . .	50
4.4.1	Detection Accuracy . . . . .	50
4.4.2	Limitations of Mobile Malicious Node Attacks . . . . .	51
4.4.3	False Positives . . . . .	55
4.5	Performance Analysis . . . . .	56
4.6	Experimental Study . . . . .	58
4.6.1	Simulation Environment . . . . .	58
4.6.2	Simulation Results . . . . .	59
4.7	Summary . . . . .	62
5.	STATIC REPLICA NODE DETECTION . . . . .	63
5.1	Overview . . . . .	63
5.2	Assumptions . . . . .	65

5.3	Scheme I: The Basic Approach . . . . .	66
5.3.1	Protocol Description . . . . .	67
5.3.2	Security Analysis . . . . .	69
5.3.3	Connectivity . . . . .	71
5.4	Scheme II: Location Claim Approach . . . . .	72
5.4.1	Protocol Description . . . . .	72
5.4.2	Security Analysis . . . . .	74
5.4.3	Communication Overhead . . . . .	77
5.4.4	Computation and Claim Storage Overhead . . . . .	79
5.4.5	Risk of Denial of Service . . . . .	80
5.5	Scheme III: Multi-Group Approach . . . . .	81
5.5.1	Protocol Description . . . . .	81
5.5.2	Security Analysis . . . . .	83
5.5.3	Communication Overhead . . . . .	83
5.5.4	Computation and Claim Storage Overhead . . . . .	85
5.6	Deployment Time Checks . . . . .	85
5.6.1	Protocol Description . . . . .	87
5.6.2	Analysis . . . . .	87
5.7	Discussion . . . . .	88
5.8	Summary . . . . .	90
6.	MOBILE REPLICA NODE DETECTION . . . . .	92
6.1	Overview . . . . .	92
6.2	Assumptions . . . . .	94
6.3	Protocol Description . . . . .	95
6.3.1	Claim Generation and Forwarding . . . . .	98
6.3.2	Detection and Revocation . . . . .	98

6.4	Security Analysis . . . . .	101
6.4.1	Detection Accuracy . . . . .	101
6.4.2	Limitations of Replica Node Attacks . . . . .	102
6.5	Performance Analysis . . . . .	117
6.5.1	Communication Overhead . . . . .	117
6.5.2	Computation and Storage Overhead . . . . .	119
6.6	Experimental Study . . . . .	120
6.6.1	Simulation Environment . . . . .	120
6.6.2	Simulation Results . . . . .	123
6.7	Summary . . . . .	130
7.	WORM PROPAGATION DETECTION . . . . .	131
7.1	Overview . . . . .	132
7.2	Assumptions . . . . .	135
7.3	Protocol Description . . . . .	136
7.4	Security Analysis . . . . .	143
7.4.1	Detection Accuracy . . . . .	143
7.4.2	Limitations on Worm Propagation . . . . .	145
7.4.3	False Positives . . . . .	146
7.5	Performance Analysis . . . . .	147
7.5.1	Average Number of Samples for Decision . . . . .	148
7.5.2	Communication and Storage Overheads . . . . .	148
7.6	Experimental Study . . . . .	149
7.6.1	Simulation Environment . . . . .	149
7.6.2	Simulation Results . . . . .	153
7.7	Summary . . . . .	157
8.	CONCLUSION . . . . .	158

REFERENCES . . . . .	159
BIOGRAPHICAL STATEMENT . . . . .	168

## LIST OF FIGURES

Figure	Page
3.1 $n^*$ vs. $P^*$ when $\alpha' = 0.01$ and $\beta' = 0.01$ , $\rho_0 = 0.1$ and $\rho_1 = 0.9$ . . . . .	30
3.2 $E[n H_1]$ vs. $\rho_0$ when $\alpha' = 0.01$ and $\beta' = 0.01$ . . . . .	34
3.3 Average number of reports vs. error rate ( $\gamma$ ). . . . .	38
3.4 Probability distribution of the number of reports. . . . .	39
4.1 Minimum fraction of neighbor contact time slots to avoid detection ( $\gamma^*$ ) vs. number of time slots ( $n$ ). . . . .	52
4.2 Maximum number of time slots waiting for the first neighbor contact ( $\theta$ ) vs. the upper threshold ( $\varphi_1$ ). . . . .	54
4.3 Average number of time slots to accept $H_0$ ( $E[n H_0]$ ) vs. the upper threshold ( $\varphi_1$ ). . . . .	57
4.4 Average number of time slots to accept $H_1$ ( $E[n H_1]$ ) vs. the upper threshold ( $\varphi_1$ ). . . . .	58
4.5 Average number of time slots vs. packet loss rate ( $\rho$ ) in the trueNegative case. . . . .	59
4.6 Average number of time slots vs. packet loss rate ( $\rho$ ) in the truePositive case. . . . .	60
4.7 Probability distribution of the number of time slots when $\rho = 0.4$ and $\tau = 10$ . . . . .	61
5.1 The number of nodes affected by replica nodes. . . . .	69
5.2 The number of nodes without connectivity. . . . .	70
5.3 The detection probability $P_d(c)$ as a function of the fraction of compromised nodes $f_c$ . . . . .	75
5.4 The detection probability $P_d(c)$ as a function of the group deployment accuracy $P_z$ . . . . .	76
5.5 Communication overhead per node. . . . .	78

5.6	Multi-group claim forwarding. . . . .	82
5.7	Communication overhead per node. . . . .	84
6.1	$\Delta R$ vs. $\Delta T$ . . . . .	108
6.2	$E[n H_0]$ vs. $\lambda_0$ when $\alpha' = 0.01$ and $\beta' = 0.01$ . . . . .	117
6.3	$E[n H_1]$ vs. $\lambda_0$ when $\alpha' = 0.01$ and $\beta' = 0.01$ . . . . .	118
6.4	Average number of claims vs. $V_{\max}$ . . . . .	121
6.5	Probability distribution of the number of claims when $\gamma = 0.1$ . . . . .	122
6.6	Average number of claims vs. $D$ when $V_{\max} = 20$ m/s. . . . .	123
6.7	Average number of claims vs. $D$ when $V_{\max} = 40$ m/s. . . . .	124
6.8	Average number of claims vs. $D$ when $V_{\max} = 60$ m/s. . . . .	125
6.9	Average number of claims vs. $D$ when $V_{\max} = 80$ m/s. . . . .	126
6.10	Average number of claims vs. $D$ when $V_{\max} = 100$ m/s. . . . .	127
6.11	Probability distribution of the number of claims when $\gamma = 0.1$ and $V_{\max} = 20$ m/s. . . . .	128
6.12	Probability distribution of the number of claims when $\gamma = 0.1$ and $V_{\max} = 100$ m/s. . . . .	129
7.1	Lower bound on worm propagation detection probability $P$ of $D$ detectors vs. $D$ detectors. . . . .	144
7.2	The effect of $\chi$ on $\tau^*$ . . . . .	145
7.3	The affects of $\gamma_0$ and $\gamma_1$ on the expected number of samples conditioned on $H_0$ and $H_1$ . . . . .	147
7.4	Cumulative number of infected nodes $I_t$ vs. time slot $t$ . . . . .	151
7.5	Average number of samples vs. pairwise infection rate $\rho$ in truePositive case. . . . .	152
7.6	Average number of time slots vs. pairwise infection rate $\rho$ in truePositive case. . . . .	153
7.7	Average number of infected nodes vs. pairwise infection rate $\rho$ in truePositive case. . . . .	154
7.8	Probability distribution of the number of samples. . . . .	155

7.9	Probability distribution of the number of time slots. . . . .	156
-----	---	-----

## LIST OF TABLES

Table	Page
5.1 Frequently used notations. . . . .	66
6.1 Parameter Values Used in Simulation Experiments. . . . .	122



## CHAPTER 1

### INTRODUCTION

In this chapter, we first describe the problem statement and the challenges for the problem. We then present our contributions and the organization of the dissertation.

#### 1.1 Problem Statement

Wireless sensor networks have recently gained much attention in the sense that they can be readily deployed for many different types of missions. In particular, they are useful for the missions that are difficult for humans to carry out. For example, they are suitable for sensing dangerous natural phenomenon such as volcano eruption, biohazard monitoring, and forest fire detection. In addition to these hazardous applications, sensor networks can also be deployed for battle field surveillance, border monitoring, nuclear and chemical attack detection, intrusion detection, flood detection, weather forecasting, traffic surveillance and patient monitoring [1]. Moreover, advances in robotics have made it possible to develop a variety of new architectures for autonomous wireless networks of sensors. Mobile sensor nodes, essentially small robots with sensing, wireless communications, and movement capabilities, are useful for tasks such as static sensor deployment, adaptive sampling, network repair, and event detection [15].

To carry out a variety of missions, the network operator deploys the base station and a set of small sensor devices in the network field. Specifically, sensor devices form ad-hoc networks, collaborate with each other to sense the phenomenon associated with

the assigned missions and then send the sensory data to the base station. The network operator obtains the mission related information by analyzing the data collected at the base station. To help sensor nodes carry out the missions efficiently and effectively, many researchers proposed a variety of the network service and communication protocols [84]. Specifically, localization, coverage, compression and aggregation protocols have been proposed for the network services. Various network protocols from physical layer to transport layer have been proposed for the communication.

Since sensor networks are often deployed in an unattended manner, most of these protocols are exposed to a variety of attacks such as denial of service attacks, routing disruption and false data injection attacks, network service disruption attacks [25, 43, 75]. To defend the wireless sensor networks against these various attacks, many schemes have been developed in the literature. For instance, secure routing schemes have been proposed to mitigate routing disruption attacks [43, 54, 59]. False data injection attacks can be mitigated by using the authentication schemes [81, 86, 91]. Secure data aggregation protocols are used to prevent attacker from disrupting aggregation [8, 18, 38, 60, 78]. Many schemes have also been proposed to protect localization and time synchronization protocols from the threat [6, 29, 40, 44, 46, 51, 66, 67].

However, most of them focus on making the protocols be attack-resilient rather than removing the source of attacks. Although attack-resiliency approach mitigates the threats on the network services and communication protocols, this approach requires substantial time and effort to continuously enhance the robustness of the protocols in accordance with the emergence of new types of attacks. Moreover, since it is hard to predict new types of attacks, the protocols will likely have resiliency only after being damaged by new types of attacks. Thus, we need to detect and revoke the sources of attacks as soon as possible to substantially reduce the costs and damages

incurred by employing attack-resilience approach. The principle sources of various attacks are compromised sensor nodes in the sense that attacker can compromise sensor nodes by exploiting the unattended nature of wireless sensor networks and thus do any malicious activities with them.

A straightforward strategy to compromise sensor node is to physically capture node and extract secret keying materials from the captured node. The keying materials make compromised nodes to seem like authorized participants in the network and thus enable adversary to leverage the compromised nodes to launch a variety of attacks. For example, he can simply monitor a significant fraction of the network traffic that would pass through these compromised nodes. Alternatively, he could jam legitimate signals from benign nodes or inject falsified data to corrupt monitoring operation of the sensors. A more aggressive attacker could undermine common sensor network protocols, including cluster formation, routing, and data aggregation, thereby causing continual disruption to the network operations.

However, it will take substantial time and effort for adversary to widely spread node compromise by only physically capturing sensor nodes. To save the time and effort incurred from using node capture attack, he may launch replica node attacks for the wide-spread node compromise. In replica node attacks, adversary captures a sensor node, creates replica nodes by copying the ID and secret keying materials of the captured node into a set of new nodes, and spreads out the replicas over the network. The replica nodes are controlled by the adversary, but have keying materials that allow them to seem like authorized participants in the network. Protocols for secure sensor network communication would allow replica nodes to create pairwise shared keys with other nodes and the base station, thereby enabling the nodes to encrypt, decrypt, and authenticate all of their communications as if they were the original captured node. In the sense that replica node attacks enable the attacker to leverage

the compromise of a single node to create widespread effects on the network, they are very dangerous to the network operations in sensor networks. Furthermore, replica node attacks are more cost-efficient than node capture attacks from the perspective that the time and effort needed to inject many replica nodes into the network is much less than the effort to compromise the equivalent number of original nodes.

However, replica node attacks also require attacker to prepare as many sensor nodes as the number of replicas that he wants to generate. Thus, the replica node attacks cost will increase with the number of deployed replicas. To reduce these costs, attacker may attempt to widely spread node compromise by adopting worm propagation attack strategy. In this attack, he captures a few nodes and forces the captured nodes to spread malicious self-propagating worm over the network, resulting in the compromise of large number of benign nodes. This is also very severe type of attack from the perspective that attacker can amplify his influence by leveraging a few captured nodes.

In this dissertation, we aim to robustly and efficiently detect and revoke the compromised nodes, prevent the wide-spread node compromise through replica node attacks and worm propagation attacks in the sensor networks, leading to remove the sources of various threats against the wireless sensor networks.

## 1.2 Challenges

A straightforward solution to stop node compromise and prevention is to prevent the adversary from extracting keying materials from sensor nodes by equipping them with tamper-resistant hardware. However, this solution may be too expensive to be practical for many applications. In addition, although tamper-resistant hardware can make it significantly harder and more time-consuming to extract keying materials from captured nodes, it may still be possible to bypass tamper resistance for a small

number of nodes in a reasonable amount of time. In this attack scenario, attacker can force a few compromised nodes to spread malicious worm over the network or generate many replica nodes of them, leading to amplify his influence on the network. We thus believe that it is necessary to develop software-based countermeasures to address this threat.

In software-based approaches, it is required to utilize as many sensor nodes as possible for robust and efficient node compromise detection and prevention. This requirement is reasonable from the perspective that it forces attacker to compromise many nodes to disrupt software-based solutions. However, it will be challenging issue how to securely and efficiently design software-based approaches under this requirement. Specifically, we need to consider the following design issues. First, node compromise detection and prevention should be done with minimal communication, computation, and storage overheads. Second, the detection and prevention schemes should be robust and highly resilient against an attacker’s attempt to break the scheme. More specifically, the schemes should work unless the attacker compromises a substantial number of nodes. Finally, there should be low false negatives and low false positives, meaning that the compromised node would be detected and prevented with high probability, benign node would be misidentified as compromised node with low probability. This is important to prevent the attacker from bypassing detection and prevention schemes, and turning them into a tool for denial of service attacks.

### 1.3 Contributions

To root out the sources of a variety of attacks in the sensor networks, we propose a framework for fast and robust detection and prevention of the wide-spread node compromise in wireless sensor networks. In the framework, we first propose node compromise detection schemes, and then propose replica node detection and

worm propagation detection schemes. Specifically, the framework is composed of five components. In the first component, we propose a reputation-based trust management scheme that is designed to achieve fast detection and revocation of a group of compromised nodes [37]. The main idea of the proposed scheme is to divide the network into a set of zones, establish and maintain trust levels for each zone, and detect untrustworthy zones and perform software attestation against nodes in untrustworthy zones to detect and revoke the compromised nodes. Through analysis and simulation, we show that our proposed scheme provides effective and robust node compromise detection and revocation capability with little overhead. However, if the attacker can move the compromised nodes by employing simple robotic platforms or move the nodes by hand, he can evade the static node compromise detection scheme in the first component. In the second component, to overcome this limitation, we propose a scheme for distributed detection of mobile malicious node attacks in static sensor networks. The key idea of our proposed scheme is to quickly detect and block mobile malicious nodes by leveraging 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 demonstrate analytically and through simulation that our proposed scheme achieves fast, effective, and robust mobile malicious node detection capability with reasonable overhead.

Although the first and the second components achieve fast node compromise detection, it cannot be directly used to stop replica node attacks. To thwart the wide-spread node compromise by replica node attacks, we propose replica detection schemes in the static and mobile sensor networks. In the third component, we detect and revoke replica nodes in static sensor networks under the assumption of group deployment knowledge by leveraging the intuition that replica nodes are deployed in

more than one location [35]. Through analysis and simulation experiments, we show that our proposed schemes achieve effective and robust replica detection capability with substantially lower communication, computation, and storage overheads than prior work in the literature.

In the fourth component, we quickly detect and revoke mobile replicas by leveraging the intuition that mobile replicas appear to move faster than benign nodes and thus highly likely exceed the predefined maximum speed [36]. We show analytically and through simulation experiments that our proposed scheme provides effective and robust mobile replica detection capability with reasonable overhead. Although these two components efficiently and effectively detect and revoke replica nodes, they do not stop the wide-spread node compromise by worm propagation attack. In the fifth component, we propose a worm propagation detection scheme to battle against this type of attack. The main idea of our proposed scheme is to quickly detect and block worm propagation by leveraging the intuition that a worm’s communication pattern is different from benign traffic. Through analysis and simulation, we demonstrate that our proposed scheme effectively and efficiently performs worm propagation detection with reasonable overhead.

## 1.4 Organization

The rest of the dissertation is organized as follows. Chapter 2 describes the technical backgrounds that help understand the dissertation well, and the related research works. Chapter 3 describes the first component of the framework, a fast zone-based node compromise detection and revocation scheme in sensor networks. Chapter 4 describes the second component of the framework, a distributed scheme for mobile malicious node detection in sensor networks. Chapter 5 describes the third component of the framework, replica detection schemes with group-deployment

knowledge in static sensor networks. Chapter 6 describes the fourth component of the framework, a fast replica detection scheme in mobile sensor networks. Chapter 7 describes the fifth component, a distributed worm propagation detection scheme in sensor networks. We conclude this disseratation in Chapter 8.



## CHAPTER 2

### BACKGROUND

In this chapter, we state the technical backgrounds that are essential to understand our research works well. We first describe the group deployment strategy that is used as a key assumption for static replica detection. We next describe the Sequential Probability Ratio Test that is used as an underlying framework for static and mobile node compromise detections, mobile replica detection, and worm propagation detection. We then describe the random mobility model under which we evaluate mobile replica detection scheme. Finally, we present the research works that are related to our proposed schemes.

#### 2.1 Group Deployment Strategy

In this strategy, sensor nodes are grouped together by the network operator and programmed with the corresponding group information before deployment, with each group of nodes being deployed towards the same location, called the *group deployment point*. After deployment, the group members exhibit similar geographic relations. We argue that this is reasonable for any sensor network in which nodes are spread over a field, such as being dropped from an airplane or spread out by hand. A simple way to do this would be to keep the groups of nodes in bags marked with the group IDs and use a marked map with the group IDs on it. All that is needed is a map of the territory and a way to pre-determine the deployment points, such as assigning a point on a grid to each group. This argument is further supported by the fact that the group deployment strategy has been used for various applications in sensor

networks such as key distribution [22, 52], detection of anomalies in localization [23], and public key authentication [24].

The deployment follows a particular probability density function (pdf), say  $f$ , which describes the likelihood of a node being a certain distance from its group deployment point. For simplicity, we use a two-dimensional Gaussian distribution to model  $f$ , as in [23]. Let  $(x_G, y_G)$  be the group deployment point for a group  $G$ . A sensor node in group  $G$  is placed in a location  $(x, y)$  in accordance with the following model:

$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_G)^2 + (y-y_G)^2}{2\sigma^2}} \quad (2.1)$$

where  $(x_G, y_G)$  is group deployment point and  $\sigma$  is the standard deviation of the two-dimensional Gaussian distribution.

According to Equation 2.1, 68% and 99% of nodes in a group are placed within a circle whose center is the group deployment point and radius is  $\sigma$  and  $3\sigma$ , respectively.

## 2.2 Sequential Probability Ratio Test

Sequential Probability Ratio Test (SPRT) is a statistical decision theory that was developed by Wald [72]. The SPRT is also known as sequential hypothesis testing in the sense that the test is sequentially performed with single or composite hypothesis. However, the SPRT is distinct from the classical hypothesis testing in the sense that the sample size is not fixed in prior to the test. In other words, the number of samples is treated as random variable in the SPRT. This feature makes the test reach a decision much earlier than the classical hypothesis testing while achieving the low false positive and negative rates. Some researchers [42, 76] also modeled the SPRT as one-dimensional random walk with lower and upper limits. Before the random walk starts, null and alternate hypotheses are defined in such a way that the null hypoth-

esis is associated with the lower limit while the alternate one is associated with the upper limit. A random walk starts from a point between two limits and moves toward the lower or upper limit in accordance with each observation. If the walk reaches (or exceeds) the lower or upper limit, it terminates and the null or alternate hypothesis is selected, respectively.

Now we describe the details of the SPRT. We adopt a simple application of the defective semiconductor equipment detection for the purpose of illustration of the SPRT. In this application, in order to detect the defective equipments, we leverage the intuition that defective equipments are more likely to produce defective semiconductor than indefective equipments. Let  $o_i$  denote the  $i$ th semiconductor produced by equipment  $E$ .  $o_i$  is considered as a sample in the SPRT. Let  $S_i$  denote a Bernoulli random variable that is defined as:

$$S_i = \begin{cases} 0 & \text{if } o_i \text{ is indefective} \\ 1 & \text{if } o_i \text{ is defective} \end{cases}$$

The success probability  $\lambda$  of Bernoulli distribution is defined as:

$$\Pr(S_i = 1) = 1 - \Pr(S_i = 0) = \lambda \quad (2.2)$$

If  $\lambda$  is smaller than or equal to a preset threshold  $\lambda'$ , it is likely that equipment  $E$  is not defective. On the contrary, if  $\lambda > \lambda'$ , it is likely that equipment  $E$  is defective. The problem of deciding whether equipment  $E$  is defective or not can be formulated as a hypothesis testing problem in which the null and alternate hypotheses are associated with  $\lambda \leq \lambda'$  and  $\lambda > \lambda'$ , respectively. In this problem, we need to devise an appropriate sampling strategy in order to prevent hypothesis testing from leading to a wrong decision. In particular, we should specify the maximum possibilities of

wrong decisions that we want to tolerate for a good sampling strategy. To do this, we reformulate the above hypothesis testing problem as one with null and alternate hypotheses of  $\lambda \leq \lambda_0$  and  $\lambda \geq \lambda_1$ , respectively, such that  $\lambda_0 < \lambda_1$ . In this reformulated problem, the acceptance of the alternate hypothesis is regarded as a false positive error when  $\lambda \leq \lambda_0$ , and the acceptance of the null hypothesis is regarded as a false negative error when  $\lambda \geq \lambda_1$ . To prevent the decision process from making these two types of errors, we define a user-configured false positive  $\alpha'$  and false negative  $\beta'$  in such a way that the false positive and negative should not exceed  $\alpha'$  and  $\beta'$ , respectively.

To understand the basis of this sampling plan, we present how the SPRT is performed to make a decision about equipment  $E$  from the  $n$  observed samples. We first denote  $H_0$  as the null hypothesis that  $E$  is indefective and  $H_1$  as the alternate one that  $E$  is defective. We then define  $L_n$  as the log-probability ratio on  $n$  samples, given as:

$$L_n = \ln \frac{\Pr(S_1, \dots, S_n | H_1)}{\Pr(S_1, \dots, S_n | H_0)} \quad (2.3)$$

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.

If equipment  $E$  is judged as indefective, we restart the SPRT with newly arrived samples. If, however,  $E$  is determined to be defective, we terminate the SPRT on  $E$  and repair  $E$ .

### 2.3 Random Mobility Model

Several random mobility models [3, 4, 5] have been proposed to evaluate the performance of protocols in mobile ad hoc networks (MANET). Among these models, the Random Waypoint Mobility (RWM) model is widely used in the literature. In the RWM model, each node moves to a randomly chosen location with a randomly selected speed between a predefined minimum and maximum speed. After reaching that location, it stays there for a predefined pause time. It then randomly chooses another location after that pause time and moves to that location. This random movement process is repeated throughout the network lifetime. However, according to [85], when the RWM model is used in simulation, it takes some time for the probability distribution of the movement of nodes to converge to a steady-state distribution after the start of simulation. Furthermore, the convergence time is changed in accordance with the parameters of the mobility model and the performance of the network. Thus, it is hard to find a steady-state distribution in the RWM model. To resolve this problem, Le Boudec et al. [3] proposed the Random Trip Mobility (RTM) model as a generic framework for finding the steady-state distribution of any mobility model based on random movement. In the RTM model, each mobile node's movement is determined in accordance with a set of trips. A path  $P$  is chosen per trip  $T$  according to a specific random mobility model. A mobile node's position at a specific time  $t$  in the  $i$ th trip  $T_i$  is determined in accordance with path  $P_i(t)$  and the fraction of elapsed time  $t$  on the  $i$ th trip.

### 2.4 Related Work

We first describe a number of research works on static node compromise detection in wireless sensor networks. Software-attestation based schemes have been

proposed to detect the subverted software modules of sensor nodes [55, 62, 64, 79]. Specifically, in [55, 62, 64], the base station checks whether the flash image codes have been maliciously altered by performing attestation randomly chosen portions of image codes or the entire codes. In [79], a sensor node’s flash image codes are attested by its neighbors. However, all these schemes require each sensor to be periodically attested and thus incur a large overhead in terms of communication and computation. Our static node compromise detection scheme does not require periodic attestation but one-time attestations against untrustworthy zones.

Reputation-based trust management schemes have been proposed to manage individual node’s trust in accordance with its actions [30, 45, 68]. Specifically, Ganeriwal et al. [30] proposed a reputation-based trust management scheme in which a Bayesian formulation is used to compute an individual node’s trust. Sun et al. [68] proposed information theoretic frameworks for trust evaluation. Specifically, entropy-based and probability-based schemes have been proposed to compute an individual node’s trust. Li et al. [45] leveraged node mobility to reduce an uncertainty in trust computation and speed up the trust convergence. However, these trust management schemes do not revoke compromised nodes and thus compromised nodes can keep performing malicious activities in the network. Our static node compromise detection scheme revokes compromised nodes in untrustworthy zones by using software-attestation technique.

ID traceback schemes have been proposed to locate the malicious source of false data [82, 89]. However, they only trace a source of the data sent to the base station and thus they do not locate the malicious sources that send false data or control messages to other benign nodes in the network. Our static node compromise detection scheme does not have this limitation.

We then describe some related works for mobile node compromise detection in sensor networks. 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 [35] that using this simple approach (their *Scheme I*) leads to substantial network connectivity problems.

Location distinction schemes [27, 47, 57, 87] 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.

Next we present the related works for replica node detection in sensor networks. The first work on replica node detection is due to Parno et al. [58], who proposed randomized and line-selected multicast schemes to detect replicas in static wireless sensor networks. In the randomized multicast scheme, every node is required to multicast a signed location claim to randomly chosen witness nodes. A witness node that receives two conflicting location claims for a node concludes that the node has been replicated and initiates a process to revoke the node. The line-selected multicast scheme reduces the communication overhead of the randomized multicast scheme by having every claim-relaying node participate in the replica detection and revocation process. Static replica detection schemes are developed for replica detection when the

nodes have access to reasonable deployment knowledge. In chapter 5, we have shown that static replica detection schemes have a number of advantages over [58].

Conti et al. [13] proposed a Randomized, Efficient, and Distributed (RED) protocol to enhance the line-selected multicast scheme of [58] in terms of replica detection probability, storage and computation overheads. However, RED still has the same communication overhead as the line-selected multicast scheme of [58]. More significantly, their protocol requires repeated location claims over time, meaning that the cost of the scheme needs to be multiplied by the number of runs during the total deployment time. Since static replica detection schemes only require claims when new nodes are deployed, they are much more efficient than RED in terms of communication overhead over the lifetime of the network. Zhu et al [90]. proposed localized multicast schemes based on the grid cell topology, where replicas are detected by letting location claim be multicasted to single cell or multiple cells. The main strength of [90] is that it achieves higher detection rates than the best scheme of [58]. However, [90] has similar communication overheads as [58] whereas static replica detection schemes work as well with much lower overheads.

Choi et al. [11] proposed a clone detection scheme in sensor networks. In this scheme, the network is considered to be a set of non-overlapping subregions. An exclusive subset is formed in each subregion. If the intersection of subsets is not empty, it implies that replicas are included in those subsets. However, this scheme allows an adversary to bypass the detection by placing replicas in such a way that any two adjacent subsets do not include them. Static replica detection schemes can effectively address this problem.

Xing et al. [77] proposed fingerprint-based replica node detection scheme. In this scheme, nodes report fingerprints, which identify a set of their neighbors, to the



base station. The base station performs replica detection by using the property that fingerprints of replicas conflict each other.

Finally, we introduce the research works in which worm propagation attacks are demonstrated to be practically mounted in sensor networks and describe the related work of worm propagation prevention and the limitation of it.

Several researchers have recently showed the feasibility of malicious code injection attacks and worm propagation attacks against sensor devices [28, 31, 32]. Specifically, Goodspeed et al. [31] proposed a way of running malicious code on the MSP430-based TelosB motes by exploiting string format vulnerabilities and buffer overflows. Gu et al. [32] showed that malicious code can be transiently executed on Mica2 motes and propagated to neighboring motes. Francillon et al. [28] demonstrated that malicious code can be permanently and remotely injected into the program memory of an Atmel AVR-based sensor devices such as the MicaZ mote. They also showed that the malicious code can be easily extended to a self-propagating worm. This work is the first to break a common belief that worm propagation is impossible on sensor devices equipped with the Harvard architecture.

To our knowledge, there is only one work for worm propagation prevention in sensor networks. Specifically, Yang et al. [80] used software diversity technique to prevent worm propagation in sensor networks. In this work, the network is divided into a set of grid cells and a group of sensors are deployed in each cell. Flash program is then installed into each node in such a way that all nodes in the same cell have the same version of flash program, but all nodes in a cell do not share the same version of flash program with other nodes in any adjacent cells. Even though attacker creates a worm by exploiting the vulnerability of a version of flash program assigned to a cell and infects the entire cell with that worm, he will highly likely fail to infect the adjacent cells with different versions of the flash program as long as these programs

have different vulnerabilities. Hence, the main strength of this work is that it prevents a worm from propagating between two adjacent cells with little overhead. However, it is hard to automatically guarantee that different versions of flash program have different vulnerabilities. If we can discover different vulnerabilities, we can fix them and thus we do not need this scheme. On the other hand, when all versions of flash program have the same vulnerability that has not yet been identified, attacker can create a worm by exploiting this vulnerability and infect all sensor nodes with it, leading to failure of this approach.

Unlike the software diversity approach, our proposed scheme focuses on the detection of worm propagation rather than passively deterring the worm from propagating and thus does not have the aforementioned limitations of [80].

## CHAPTER 3

### ZONETRUST: ZONE-BASED NODE COMPROMISE DETECTION

In this chapter, we propose a fast zone-based node compromise detection and revocation scheme in sensor networks. The main idea of the proposed scheme is to use the sequential hypothesis testing to detect suspect regions in which compromised nodes are likely placed. In these suspect regions, the network operator performs software attestation against sensor nodes, leading to the detection and revocation of the compromised nodes. We show analytically and through simulation experiments that the proposed scheme provides effective and robust node compromise detection and revocation capability with little overhead.

#### 3.1 Overview

In this section, we present an overview of a zone-based node compromise detection and revocation scheme.

To mitigate the node compromise attack, several researchers have proposed various node compromise detection schemes in the context of wireless networks [30, 45, 55, 62, 64, 68, 79]. Reputation-based trust management schemes have been proposed to manage an individual node's trust in accordance with its activities [30, 45, 68]. However, although malicious nodes can be identified in these schemes, they are not easily revoked due to the risk of false positives. Software-attestation based schemes have also been proposed to attest flash image codes and detect subverted image codes of compromised sensor nodes [55, 62, 64, 79]. Although these schemes achieve high node compromise detection capability, they require each sensor node to be periodi-

cally attested, as it cannot be predicted when the attacker will compromise sensors. This periodic attestation will incur substantial overhead in terms of computation and communication.

Our work is motivated from mitigating the limitations of previous schemes. In particular, we propose a reputation-based trust management scheme that is designed to facilitate fast detection and revocation of compromised nodes. The key idea of our scheme is to detect untrustworthy *zones* and perform software attestation against nodes in these zones to detect and revoke the ones that are compromised. Specifically, we first divide the network into a set of zones, establish *trust levels* for each zone, and detect untrustworthy zones by using the Sequential Probability Ratio Test (SPRT) [72]. The SPRT decides a zone to be untrustworthy if the zone’s trust is continuously maintained at low level or is quite often changed from high level to low level. Once a zone is determined to be untrustworthy, the network operator performs software attestation against all nodes in the untrustworthy zone, detects compromised nodes with subverted software modules, and physically revokes them.

The main benefit of this zone-based detection approach lies in achieving fast node compromise detection and revocation while saving the large amount of time and effort that would be incurred from using periodic software attestation. By detecting an entire zone at once, the system can identify the approximate source of bad behavior and react quickly, rather than waiting for a specific node to be identified. Also, when multiple nodes are compromised in one zone, they can all be detected and revoked at one time.

Our approach is also robust. We show in simulation and analysis that, as long as more than 50% of the nodes in each zone are honest, our scheme will quickly identify untrustworthy zones with low error rates. While using other reputation-based systems to revoke misbehaving nodes can suffer from false positives, our zone-based approach

has each zone report on itself. If nodes in the zone attempt to raise a false positive, it will only accelerate their own detection. Also, the costs of a false positive due to errors are limited to a single round of software attestation within the zone.

The rest of chapter is organized as follows. Section 3.2 presents the underlying assumptions of zone-based detection scheme. Section 3.3 describes our compromised node detection and revocation scheme using the SPRT. Section 3.4 presents the security analysis of the proposed scheme. Section 3.5 presents the performance analysis of the proposed scheme. Section 3.6 presents the results of simulations we conducted to evaluate the scheme. Finally, Section 3.7 summarizes the chapter.

## 3.2 Assumptions

In this section, we describe the network and attacker assumptions of a zone-based node compromise detection and revocation scheme.

We assume a two-dimensional *static* sensor network in which sensor nodes do not change their locations after deployment. We also assume that all direct communication links between sensor nodes are bidirectional. This communication model is common in the current generation of sensor networks.

We assume that an adversary may attempt to find as many different regions as possible and compromise a set of sensor nodes in each region. However, we place limits on his attack capability such that he does not compromise a majority of the nodes in each region. This is reasonable, since compromising a majority of sensor nodes in a region is far from optimal. This is mainly because his influence is limited to the region while he spends substantial time and effort to compromise many nodes. The same time and effort could instead be used to spread out compromised nodes over a wider area and cause greater disruption to the network. Moreover, it is worth noting that the adversary can easily defeat most node compromise detection schemes

if he could compromise a major fraction of the region. This is because a large number of compromised nodes in a region can prevent relatively small number of benign nodes from performing node compromise detection in the region.

### 3.3 Protocol Description

In this section, we present the details of our technique to detect and revoke compromised nodes in zones.

As described in Section 3.1, reputation-based trust management schemes do not stop compromised nodes doing malicious activities in the network. Also, the existing schemes based on software attestation require each sensor to be periodically attested because it cannot be predicted when attacker compromises sensors. The periodic attestation of individual nodes will incur large overhead in terms of computation and communication. To mitigate the limitations of both approaches, we propose a zone-based node compromise detection and revocation scheme in a two-dimensional *static* wireless sensor network, where the locations of sensor nodes do not change after deployment. Our scheme facilitates node compromise detection and revocation by leveraging zone trust information. Specifically, we divide the network into a set of zones, establish trust per zone, and detect untrustworthy zones in accordance with zone trust values. Once a zone is determined to be untrustworthy, the network operator attests the software modules of all sensors in the untrustworthy zone, and detects and revokes compromised nodes in that zone.

A straightforward approach for untrustworthy zone detection is to decide a zone as untrustworthy by observing a single evidence that its trust value is less than a pre-defined threshold. However, this approach does not consider the zone trust measurement error. Due to the error occurrence in the zone trust measurement, trustworthy (resp. untrustworthy) zone could be detected as untrustworthy (resp. trustworthy).

To minimize these false positive and negatives, we need to make a decision with multiple pieces of evidence rather than a single evidence. To meet this need, we propose a zone-based node compromise detection and revocation scheme based on the Sequential Probability Ratio Test (SPRT) [72]. We believe that the SPRT is well-suited for tackling untrustworthy zone detection problem in the sense that we can construct a random walk with two limits in such a way that each walk is determined by the trust value of a zone; the lower and upper limits are properly configured to be associated with the excess and shortfall of a predefined trust threshold, respectively.

We apply the SPRT to node compromise detection and revocation problem as follows. Every sensor node in a zone acts as *trust aggregator* in a round robin manner. In each time slot, the trust aggregator computes a trust level for its zone and reports the zone's trust level to the base station. The base station performs the SPRT with zone trust information. Each time a zone's trust is below (resp. above or equal to) a preset trust threshold, it will expedite the random walk to hit or cross the upper (resp. lower) limit and thus lead to the base station accepting the alternate (resp. null) hypothesis that a zone is untrustworthy (resp. trustworthy). Once the base station decides that a zone is untrustworthy, the network operator performs software attestations against all sensor nodes to detect and revoke the compromised nodes in the zone.

Specifically, the network operator assigns a unique ID to every sensor node and divides the network into  $z$  non-overlapping zones before deployment. A zone can have any type of shape. He also pre-loads secret keying materials onto each sensor node for pairwise key establishment; we can use any key pre-distribution technique for sensor networks, such as [7, 17, 21, 26, 88].

The network operator also pre-loads a shared secret key with the base station onto each node. We also assume that every sensor node is able to obtain its loca-

tion information and identify its placement zone by using existing secure localization schemes such as [6, 46, 51]. Finally, we assume time synchronization such that the clocks of all nodes in a zone are loosely synchronized with a maximum error of  $\epsilon$ . This assumption can be easily achieved by using the existing secure time synchronization methods [29, 40, 66, 67]. This assumption will incur little overhead since the time synchronization protocol only needs to be used in the zone. Our proposed protocol proceeds in three phases.

### 3.3.1 Phase I: Zone Discovery and Trust Aggregator Selection

After deployment, every sensor node  $u$  finds out its location and determines the zone to which it belongs. We call this zone the *home zone*. From  $u$ 's point of view, we call other zones the *foreign zones*. Node  $u$  discovers every other node residing in the same zone. After the zone discovery process, the Trust Aggregator (TA) is selected in a round robin manner. Specifically, the time domain of a zone is partitioned into time slots. An initial duty time slot is assigned to each node  $u$  in the zone according to the ascending order of the nodes' IDs. Each node  $u$  then acts as trust aggregator every  $S$  time slots starting from its initial duty time slot, where  $S$  is the number of nodes residing in the zone.

### 3.3.2 Phase II: Trust Formation and Forwarding

For each time slot  $T_i$ , each node  $u$  in zone  $Z$  computes *neighborhood-trust* that is defined in accordance with the difference between the probability distributions of the information generated by  $u$  and the information sent to  $u$  by  $u$ 's neighboring nodes in zone  $Z$ . Neighborhood-trust acts as an indicator of how much information is shared by two neighboring nodes. The more information  $u$  shares with its neighbors, the



more belief  $u$  has in its neighbors. To compute the difference between two probability distributions, we use the information-theoretic metric KL-divergence that is known to be suited for this computation [14]. Specifically, let us assume that  $u$ 's generated information follows a probability distribution with mean  $\mu$  and standard deviation  $\sigma$  and falls within the range  $[\mu - c\sigma, \mu + c\sigma]$  with probability  $p$ , where  $c$  is constant value. Moreover, we assume that the information sent to  $u$  by  $u$ 's neighbors falls within the range  $[\mu - c\sigma, \mu + c\sigma]$  with probability  $q$ . Node  $u$  computes KL-divergence  $D$  as follows:

$$D = p \times \ln \frac{p}{q} + (1 - p) \times \ln \frac{1 - p}{1 - q}$$

It then computes neighborhood-trust  $n_u = \frac{1}{1+D}$ . The definition of neighborhood-trust is reasonable in the sense that neighborhood-trust is inversely proportional to  $D$  and thus it will be one if two probability distributions exactly match with each other. Let node  $v$  be the TA in time slot  $T_i$ . Node  $u$  sends  $v$  a neighborhood-trust report that is defined as  $\{u||n_u||MAC_{K_{uv}}(u||n_u)\}$ , where MAC stands for Message Authentication Code and  $K_{uv}$  is the shared secret key between  $u$  and  $v$ . Upon receiving a neighborhood-trust report from  $u$ , node  $v$  verifies the authenticity of  $u$ 's neighborhood-trust report with  $K_{uv}$  and discards the report if it is not authentic.  $v$  collects the neighborhood-trust reports that were measured during  $T_i$  from all nodes in zone  $Z$  and aggregates the received neighborhood-trusts by using the average function. We call the aggregated version of neighborhood-trusts the *zone-trust*.  $v$  sends the base station  $Z$ 's zone-trust report, defined as  $\{v||s_v||Z||t||MAC_{K_v}(v||s_v||Z||t)\}$ , where  $s_v$  is a timestamp indicating the generation time of report,  $t$  is  $Z$ 's zone-trust and  $K_v$  is the shared secret key between  $v$  and the base station.

### 3.3.3 Phase III: Detection and Revocation

Upon receiving a zone-trust report from a TA in zone  $Z$ , the base station verifies the authenticity of TA's report with the secret shared key between TA and itself and the freshness of the timestamp, and discards the report if it is not authentic or does contain stale timestamp. The base station also maintains a record per TA associating each TA's ID with its home zone and timestamp. This prevents compromised TAs from claiming multiple home zones and from launching replay attacks with benign zone-trust reports. We denote the authentic reports from TAs in zone  $Z$  by  $R_1, R_2, \dots$ . The base station extracts zone trust information  $t_i$  from report  $R_i$ . Let  $\tau$  be a predefined trust threshold and  $B_i$  be denote a Bernoulli random variable that is defined as:

$$B_i = \begin{cases} 1 & \text{if } t_i < \tau \\ 0 & \text{if } t_i \geq \tau \end{cases}$$

The success probability  $\rho$  of Bernoulli distribution is defined as  $\rho = \Pr(B_i = 1) = 1 - \Pr(B_i = 0)$ . If  $\rho$  is smaller than or equal to a preset threshold  $\rho'$ , it is likely that zone  $Z$  is trustworthy. On the contrary, if  $\rho > \rho'$ , it is likely that zone  $Z$  is untrustworthy. The problem of deciding whether  $Z$  is trustworthy or not can be formulated as a hypothesis testing problem with null and alternate hypotheses of  $\rho \leq \rho_0$  and  $\rho \geq \rho_1$ , respectively, such that  $\rho_0 < \rho_1$ . We define user-configured false positive rate  $\alpha'$  and false negative rate  $\beta'$  in order to provide the upper bounds of false positives and false negatives in the hypothesis testing problem.

Now we present how the SPRT is used to make a decision about zone  $Z$  from the  $n$  observed samples, where trust information  $t_i$  is treated as a sample. Let us define  $H_0$  as the null hypothesis that zone  $Z$  is trustworthy and  $H_1$  as the alternate

hypothesis that zone  $Z$  is untrustworthy. We then define  $L_n$  as the log-probability ratio on  $n$  samples, given as:

$$L_n = \ln \frac{\Pr(B_1, \dots, B_n | H_1)}{\Pr(B_1, \dots, B_n | H_0)}$$

Assume that  $B_i$  is independent and identically distributed. Then  $L_n$  can be rewritten as:

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

Let  $\omega_n$  denote the number of times that  $B_i = 1$  in the  $n$  samples. Then we have

$$L_n = \omega_n \ln \frac{\rho_1}{\rho_0} + (n - \omega_n) \ln \frac{1 - \rho_1}{1 - \rho_0}$$


where  $\rho_0 = \Pr(B_i = 1 | H_0)$ ,  $\rho_1 = \Pr(B_i = 1 | H_1)$ . The rationale behind the configuration of  $\rho_0$  and  $\rho_1$  is as follows:  $\rho_0$  should be configured in accordance with the likelihood of the occurrence that trustworthy zone is determined to have low trust value due to neighborhood-trust measurement error;  $\rho_1$  should be configured to consider the likelihood of the occurrence that an untrustworthy zone is determined to have low trust value.

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

- $\omega_n \leq \lambda_0(n)$  : accept  $H_0$  and terminate the test.
- $\omega_n \geq \lambda_1(n)$  : accept  $H_1$  and terminate the test
- $\lambda_0(n) < \omega_n < \lambda_1(n)$  : continue the test process with another observation.

Where:

$$\lambda_0(n) = \frac{\ln \frac{\beta'}{1-\alpha'} + n \ln \frac{1-\rho_0}{1-\rho_1}}{\ln \frac{\rho_1}{\rho_0} - \ln \frac{1-\rho_1}{1-\rho_0}}, \quad \lambda_1(n) = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\rho_0}{1-\rho_1}}{\ln \frac{\rho_1}{\rho_0} - \ln \frac{1-\rho_1}{1-\rho_0}}$$

 If a zone  $Z$  is judged as trustworthy, the base station restarts the SPRT with newly arrived zone-trust reports. If, however,  $Z$  is determined to be untrustworthy, the base station terminates the SPRT on  $Z$ , and the network operator detects and revokes the compromised nodes by sending the mobile robots to  $Z$  and making them perform software attestation against sensor nodes in zone  $Z$ . We can use the software attestation techniques that have been proposed in [55, 62, 64, 79]. The main idea of these attestation techniques is to detect the subverted parts in the flash image codes by checking whether the tested image codes match with the original image codes.

### 3.4 Security Analysis

In this section, we will first describe the detection accuracy of our proposed scheme and then discuss possible attack approaches and the defense strategies against them.

#### 3.4.1 Detection Accuracy

In the SPRT, the following types of errors are defined.

- $\alpha$  : error probability that the SPRT leads to accepting  $H_1$  when  $H_0$  is true.
- $\beta$  : error probability that the SPRT leads to accepting  $H_0$  when  $H_1$  is true.

Since  $H_0$  is the hypothesis that a zone  $Z$  is trustworthy,  $\alpha$  and  $\beta$  are the false positive and false negative probabilities of the SPRT, respectively. According to Wald's theory [72], the upper bounds of  $\alpha$  and  $\beta$  are:

$$\alpha \leq \frac{\alpha'}{1 - \beta'}, \quad \beta \leq \frac{\beta'}{1 - \alpha'} \quad (3.1)$$

Furthermore, Wald proved that the sum of the false positive and negative probabilities of the SPRT are limited by the sum of user-configured false positive and negative probabilities. Namely, the following inequality holds:

$$\alpha + \beta \leq \alpha' + \beta' \quad (3.2)$$

Since  $\beta$  is the false negative probability,  $(1 - \beta)$  is the untrustworthy zone detection probability. Accordingly, the lower bound on the untrustworthy zone detection probability will be:

$$(1 - \beta) \geq \frac{1 - \alpha' - \beta'}{1 - \alpha'} \quad (3.3)$$

From Equations 3.1 and 3.3, we can see that low user-configured false positive and negative probabilities will lead to a low false negative probability for the sequential test process. Hence, it will result in high detection rates. For instance, if the user configures  $\alpha'$  and  $\beta'$  to both be 0.01, then untrustworthy zone detection is guaranteed with probability 0.99.

#### 3.4.2 Limitations of Node Compromise Attacks

We first consider the *false zone-trust report attack* in which compromised TAs report false zone-trust values to the base station. This attack can take two forms. First, the compromised TAs could send reports of low zone-trust values to the base station when the zone-trust is actually high. Since this attack leads to quicker detection of the compromised TAs, the attacker will not benefit from this approach. Second, the compromised TAs could send reports of high zone-trust values to the base station when the zone-trust is actually low. In this type of attack, the attacker will have all compromised TAs report a zone-trust of 1.0 to the base station to prevent the untrustworthy zone from being detected. We investigate the impact of this

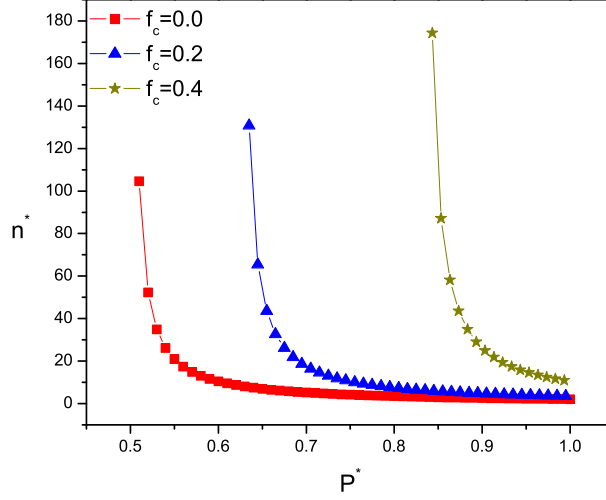


Figure 3.1:  $n^*$  vs.  $P^*$  when  $\alpha' = 0.01$  and  $\beta' = 0.01$ ,  $\rho_0 = 0.1$  and  $\rho_1 = 0.9$ .

attack on the detection capability of our scheme. For this investigation, we look into the impact of the fraction of compromised nodes in a zone on the detection capability of our scheme. Let us denote  $n$  and  $f_c$  by the number of samples and the fraction of compromised nodes in a zone, respectively. Since all compromised TAs generate a zone-trust of 1.0,  $\omega_n$  will be  $(1 - f_c) \sum_{i=1}^n Pr(B_i = 1)$ . The following Inequality also holds:

$$\omega_n = (1 - f_c) \sum_{i=1}^n Pr(B_i = 1) \geq n(1 - f_c)P^* \geq \lambda_1(n) \quad (3.4)$$

Where  $P^*$  is the minimum value of  $Pr(B_i = 1)$ . Let us denote  $n^*$  as the minimum value of the number of samples  $n$  for Inequality 3.4 to hold. From Inequality 3.4, we derive  $n^*$  as follows:

$$n^* = \frac{\ln \frac{1-\beta'}{\alpha'}}{(\ln \frac{\rho_1}{\rho_0} - \ln \frac{1-\rho_1}{1-\rho_0})(1-f_c)P^* - \ln \frac{1-\rho_0}{1-\rho_1}}$$

Where  $\frac{\ln \frac{1-\rho_0}{1-\rho_1}}{(\ln \frac{\rho_1}{\rho_0} - \ln \frac{1-\rho_1}{1-\rho_0})(1-f_c)} < P^* \leq 1$ . This restriction of  $P^*$  is required to detect untrustworthy zones in spite of the false zone-trust report attack. In other words, if

this restriction of  $P^*$  does not hold, our scheme can be breached by a false zone-trust report attack.

Now we study analytically how  $P^*$  and  $f_c$  affect  $n^*$  as shown in Figure 3.1. For this study, we set  $\alpha' = 0.01$ ,  $\beta' = 0.01$  and  $\rho_0 = 0.1$ ,  $\rho_1 = 0.9$ . In this configuration,  $P^* = 1$  holds when  $f_c = 0.5$ . This means that our scheme will fail to detect untrustworthy zones under false zone-trust report attack if more than half of nodes in a zone are compromised. In other words, our scheme is resilient against false zone-trust report attack as long as the fraction of compromised nodes is at most 50%. Specifically, the false zone-trust report attack will fail under the circumstances that  $P^* \geq 0.51, 0.635, 0.843$  when  $f_c = 0.0, 0.2, 0.4$ , respectively, as shown in Figure 3.1. We also see that  $n^*$  tends to decrease as  $P^*$  increases in all cases of  $f_c$ . This means that a high value of  $P^*$  helps the SPRT to accept  $H_1$  with a small number of samples. For a given value of  $P^*$ , we observe that a small value of  $n^*$  is achieved for low values of  $f_c$ . This indicates that the SPRT accepts  $H_1$  in fewer samples as the fraction of compromised nodes in a zone decreases.

Another important attack to address is the *reporting rule violation attack* in which compromised TAs in a zone violate the rule of reporting zone-trust in only duty time slots and send fake zone-trust reports to the base station at any time. In particular, if the attacker forces compromised TAs to report fake zone-trusts to the base station at very fast rate, it will expedite the base station to make a wrong decision in accordance with fake zone-trust values even though the base station receives genuine zone-trust reports from benign TAs in their correct duty time slots. Although fake zone-trust values can be high or low, the attacker will be interested in fake reports with high zone-trust values that help the zone appear to be trustworthy when the zone is actually untrustworthy. Note that this attack will be beneficial to attacker only when he compromises a small fraction of nodes in a zone. This is because he

can use the false zone-trust report attack if he compromises the majority of nodes in a zone. Accordingly, we assume that the majority of nodes in a zone are benign and they conform to the zone-trust reporting rule. Given this assumption, a straightforward countermeasure against this attack is to let the base station compute the delay between every two consecutive zone-trust reports from a TA  $v$  in a zone and check whether the difference between the delay for  $v$  and the average delay for all TAs in a zone is more than a predefined threshold. If so, it is highly likely that  $v$  violates the round-robin zone-trust report process and is therefore considered to be compromised.

To more quickly decide whether a TA violates the zone-trust reporting rule, we can replace the static threshold approach with the SPRT as follows. Let  $\xi_v$  be denote a zone-trust report delay for TA  $v$  in zone  $Z$  measured by the base station. Moreover, let us define  $\mu'$  and  $\sigma'$  as the mean and standard deviation of the report delays for all TAs in zone  $Z$  and let  $I_v$  be a Bernoulli random variable defined as:

$$I_v = \begin{cases} 0 & \text{if } |\xi_v - \mu'| < \theta\sigma' \\ 1 & \text{if } |\xi_v - \mu'| \geq \theta\sigma' \end{cases}$$

We need to configure  $\theta$  in accordance with the likelihood of occurrence of  $I_v = 1$ . We obtain this likelihood from Chebyshev's Inequality:

$$\Pr(|X - E[X]| \geq a) \leq \frac{V[X]}{a^2}, \text{ for any } a > 0 \quad (3.5)$$

where  $X$  is a random variable and  $E[X]$  and  $V[X]$  are the mean and variance of  $X$ , respectively. We can easily obtain from Inequality 3.5:

$$\Pr(I_v = 1) = \Pr(|\xi_v - \mu'| \geq \theta\sigma') \leq \frac{1}{\theta^2}$$



for any  $\theta > 1$ . Hence,  $\theta$  should be configured to hold  $1 < \theta \leq \frac{1}{\sqrt{\Pr(I_v=1)}}$ .

Now we define  $H_0$  and  $H_1$  as follows:  $H_0$  is the hypothesis that  $v$  conforms to the zone-trust reporting rule, and  $H_1$  is the hypothesis that  $v$  does not conform to the zone-trust reporting rule. For each measured delay for  $v$ , the base station performs the SPRT with the above definitions. If it accepts  $H_1$ , the network operator detects and revokes the compromised nodes in the zone to which  $v$  belongs. Otherwise, it continues the SPRT with newly measured delays for  $v$ .

Finally, the attacker can launch a *foreign zone attack* in which compromised nodes do not perform malicious activities in their home zone  $Z$  but in a foreign zone  $Z'$  that falls within their communication range. In this attack, the compromised nodes do not participate in the zone-trust report process to conceal their locations from the base station. Accordingly, even if the base station decides that the foreign zone  $Z'$  is untrustworthy and the network operator performs software attestation against the foreign zone, it will fail to detect these compromised nodes since they are not in  $Z'$ . Hence, a set of compromised nodes in zone  $Z$  can avoid being detected while they maliciously impact foreign zone  $Z'$ . To defend against this attack, the base station first makes a list of the nodes residing in zone  $Z'$  by recording the IDs of the TAs when it receives their zone-trust reports. It then sends this list to every node on the list. Each node  $u$  in zone  $Z'$  computes the individual trust on every node  $v$  that is not included in this list while acting as  $u$ 's neighbor. We can use previously-proposed schemes [29, 68] for the individual node trust computation. If  $u$  decides that  $v$  is untrustworthy, it halts all communication with  $v$ . Accordingly, compromised node  $v$  will fail to affect the foreign zone  $Z'$  as well as its home zone  $Z$ .

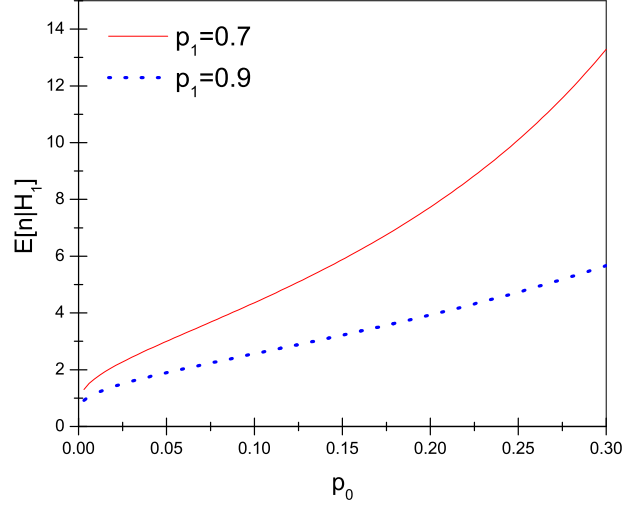


Figure 3.2:  $E[n|H_1]$  vs.  $\rho_0$  when  $\alpha' = 0.01$  and  $\beta' = 0.01$ .

### 3.5 Performance Analysis

In this section, we analyze the performance of our scheme in terms of communication, computation, and storage overheads.

#### 3.5.1 Communication Overhead

Let us first describe how many observations are required on average for the base station to decide whether a zone is trustworthy or not. Then we will present the communication overhead of our scheme.

Let  $n$  denote the number of samples to terminate the SPRT. Since  $n$  varies with the types of samples, it is treated as a random variable with an expected value  $E[n]$ . According to [72],  $E[n]$  is obtained as follows:

$$E[n] = \frac{E[L_n]}{E \left[ \ln \frac{\Pr(B_i|H_1)}{\Pr(B_i|H_0)} \right]} \quad (3.6)$$

From this equation, we compute the expected values of  $n$  conditioned on hypotheses  $H_0$  and  $H_1$  as follows:

$$E[n|H_0] = \frac{(1 - \alpha') \ln \frac{\beta'}{1-\alpha'} + \alpha' \ln \frac{1-\beta'}{\alpha'}}{\rho_0 \ln \frac{\rho_1}{\rho_0} + (1 - \rho_0) \ln \frac{1-\rho_1}{1-\rho_0}}$$

$$E[n|H_1] = \frac{\beta' \ln \frac{\beta'}{1-\alpha'} + (1 - \beta') \ln \frac{1-\beta'}{\alpha'}}{\rho_1 \ln \frac{\rho_1}{\rho_0} + (1 - \rho_1) \ln \frac{1-\rho_1}{1-\rho_0}} \quad (3.7)$$

Figure 3.2 shows how  $E[n|H_1]$  is affected by the values of  $\rho_0$  and  $\rho_1$ . Specifically,  $E[n|H_1]$  tends to increase in proportion to  $\rho_0$  when  $\rho_1$  is fixed to 0.7 and 0.9, respectively. This implies that a small value of  $\rho_0$  helps detect untrustworthy zones with a small number of reports. For a given value of  $\rho_0$ ,  $E[n|H_1]$  for the case of  $\rho_1 = 0.7$  is larger than the corresponding value for the case of  $\rho_1 = 0.9$ . This means that large values of  $\rho_1$  reduce the number of reports required for untrustworthy zone detection.

Now we compute the communication overhead of our scheme. We define communication overhead as the average number of zone-trust reports that are sent or forwarded by nodes in the network. Assume that the network is partitioned into  $z$  zones. The base station receives at most  $z$  zone-trust reports per time slot. Since the average hop distance between two randomly chosen nodes is given by  $O(\sqrt{N})$  [58] where  $N$  is the total number of sensor nodes, the communication overhead will be at most  $O(z \times \sqrt{N})$  per time slot.

### 3.5.2 Computation and Storage Overhead

We define computation and storage overhead as the average number of Message Authentication Codes (MACs) that are generated and verified by a node and the average number of zone-trust reports that need to be stored by a node, respectively. Assume that there are  $b$  nodes on average within a zone. In a zone, every node

acts as the TA in its designated time slot while acting as a zone member in other time slots. For each time slot, each zone member and each TA generate a MAC of report when sending neighborhood-trust and zone-trust reports to the TA and the base station, respectively. Also, the TA performs  $b - 1$  MAC verifications for each time slot. Thus,  $b$  nodes perform  $2b - 1$  MAC generations and verifications every time slot. Accordingly, the computation overhead per node will be  $O(1)$  on average every time slot. The base station will perform  $z$  MAC verifications every time slot because  $z$  TAs report their zone-trusts each time slot. In the SPRT, the base station does not need to keep the zone-trust report once obtaining trust information from the report. The sensor nodes also do not need to keep any reports. However, the base station needs to store timestamp information of the report to prevent replay attacks, leading to  $O(N)$  timestamp storage overhead.

### 3.6 Experimental Study

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

#### 3.6.1 Simulation Environment

We developed a simple simulation program to evaluate the proposed scheme. We simulate data generation and exchange by having nodes randomly generate data and exchange them with other nodes in the same zone. Every benign sensor node uses the normal distribution for generating data. Specifically, we first set the global data mean  $\mu = 100$ , mean variation variable  $\chi = 5$ , and standard deviation  $\sigma = 5$ . Each benign sensor node  $v$  then selects local data mean  $\mu_v$  uniformly at random from the range  $[\mu - \chi, \mu + \chi]$  and generates data in accordance with normal distribution  $N(\mu_v, \sigma)$ . Compromised nodes generate data from a range that excludes all points in

$[\mu - \chi, \mu + \chi]$ . We place 100 nodes in a zone. Each benign node and each compromised node determine the number of data values to generate uniformly at random from the range  $[0, 1000]$  and  $[0, 5000]$ , respectively. We set  $c = 1$  and  $p = 0.68$  to compute the KL-divergence described in Section 3.3.

We also set the both the user-configured false positive threshold  $\alpha$  and the false negative threshold  $\beta$  to 0.01, and we set  $\rho_0$  and  $\rho_1$  to 0.1 and 0.9, respectively. The rationale behind these configurations is discussed in Section 3.5. To emulate the zone-trust measurement errors caused by the inaccuracy of neighborhood-trust measurement, we modify the measured zone-trusts with error rate  $\gamma$ . Specifically, we take zone-trust  $t$  measured using perfect neighborhood-trust measurement and generate zone-trust  $t'$  selected uniformly at random from the range  $[t - t\gamma, t + t\gamma]$ . We configured a preset trust threshold  $\tau$  in inversely proportional to  $\gamma$  as follows:  $\tau = 0.9$  when  $\gamma = 0.05, 0.1$ ,  $\tau = 0.85$  when  $\gamma = 0.15$ , and  $\tau = 0.8$  when  $\gamma = 0.2$ . For each case, we set  $f_c$  to the values of 0.0, 0.2, 0.4. These configurations are reasonable because the rise of error rate likely results in the decrease of zone-trust values. A zone is regarded as trustworthy when  $f_c = 0.0$  and untrustworthy when  $f_c = 0.2, 0.4$ .

### 3.6.2 Simulation Results

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

- *Number of Reports* is the number of zone-trust reports required for the base station to decide whether a zone is trustworthy or not.
- *False Positive* is the error probability that a trustworthy zone is misidentified as an untrustworthy zone when  $f_c = 0.0$ .
- *False Negative* is the error probability that an untrustworthy zone is misidentified as a trustworthy zone when  $f_c = 0.2, 0.4$ .

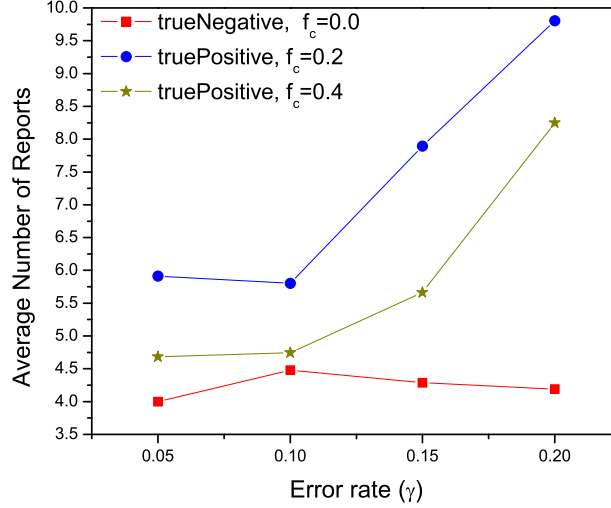


Figure 3.3: Average number of reports vs. error rate ( $\gamma$ ).

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 the zone is untrustworthy. We investigate the false positive and false negative rates and the number of reports.

First, we found that both false positive and false negative rates were below 0.01 for all values of  $\gamma$ . Specifically, there were no observed false negatives for any value of  $\gamma$ , and the highest false positives were measured as 0.005 when  $\gamma = 0.15$ . Thus, untrustworthy zones were always detected and trustworthy zones were misidentified as untrustworthy with at most probability of 0.005.

Second, the results of the average number of reports are shown in Figure 3.3. We present the results for two cases. One case is that there are no compromised nodes ( $f_c = 0.0$ ) in a zone and the SPRT decides that this zone is trustworthy. We call this case *trueNegative*. The other case is that there are compromised nodes ( $f_c =$

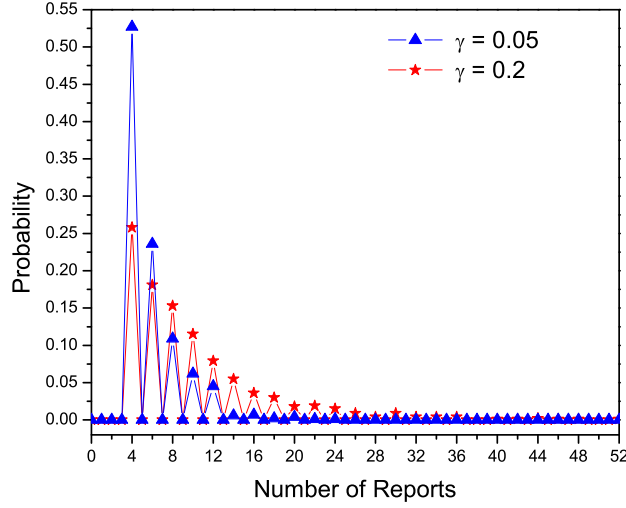


Figure 3.4: Probability distribution of the number of reports.

0.2, 0.4) in a zone, and the SPRT decides that this zone is untrustworthy. We call this case *truePositive*.

In the trueNegative case, the average number of reports reaches its maximum of 4.48 when  $\gamma = 0.1$  and  $f_c = 0.0$ . In the truePositive case, the average number of reports reaches its maximum of 9.804 when  $\gamma = 0.2$  and  $f_c = 0.2$ . Thus, the base station reaches correct decisions with a few reports in both cases. Moreover, we see that the average at low  $f_c$  is larger than that at high  $f_c$  in truePositive case. This indicates that a rise in the fraction of compromised nodes in a zone leads to an increase the chance that false data is generated in a zone and thus expedites moving the test toward  $H_1$ . We also observe that the average number of reports tends to increase with the rise of  $\gamma$  in the truePositive case. This means that an increase in the neighborhood-trust measurement error rate leads to a rise in the average number of reports.

Finally, Figure 3.4 shows the probability distribution of the number of reports in the case of truePositive when  $f_c = 0.2$ . For this distribution, we examine two

scenarios of  $\gamma = 0.05$  and  $\gamma = 0.2$ . A total of 76.3% and 70.7% of the cases fall in the range from 4 to 6 and from 4 to 10 reports in the case of  $\gamma = 0.05$  and  $\gamma = 0.2$ , respectively. This implies that in most cases, the number of reports is less than or close to the average and thus the SPRT detects untrustworthy zones with at most six and ten reports in most cases when  $\gamma = 0.05$  and  $\gamma = 0.2$ , respectively.

These results provide further evidence that compromise of a majority of the nodes in a zone will be challenging for the attacker. The attacker must be fast to compromise so many nodes in one zone without detection; compromise of fewer than half of the nodes in a zone will lead to detection in just a few reports in most cases. The network operator could use the expected minimum time to compromise a node, along with the number of nodes per zone, to help set a length for the period of time between reports. If the estimates are accurate, this would prevent the attacker from compromising the majority of nodes in a zone undetected.

### 3.7 Summary

In this chapter, we have proposed a zone-based node compromise detection and revocation scheme using the Sequential Probability Ratio Test (SPRT). Furthermore, we have described several possible attacks against the proposed scheme and proposed countermeasures against these attacks. We evaluated the scheme in simulation under various scenarios. Our experimental results show that our scheme quickly detects untrustworthy zones with a small number of zone-trust reports.



## CHAPTER 4

### MOBILE MALICIOUS NODE DETECTION

In Chapter 3, we proposed a zone-based node compromise detection scheme to detect and revoke static malicious nodes. However, 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 zone-based detection scheme. To resolve this problem, 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 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.

#### 4.1 Overview

In this section, we present an overview of the mobile malicious node detection scheme.

Prior work has investigated the detection of malicious nodes through techniques such as traceback [69, 82] and local misbehavior detection [10, 30, 65, 68]. More specifically, traceback techniques can be used to find the location of the malicious nodes given enough attack packets [69, 82]. 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 [10, 30, 65, 68]. 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.

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 can be undermined, however, by *mobile malicious nodes*. Although we only consider static sensor networks in this chapter, 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 [15, 61], 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 send flooding packets to the base station at each location. The 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 [69, 82] and other network-level defenses [19, 74] 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 [10, 65], 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 more dangerous impact on the network than regular malicious node attacks.

In this chapter, 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. In section 4.2, we describe assumptions of the network and the attackers' capabilities using mobile malicious nodes. In section 4.3, we propose a distributed detection scheme to identify and block mobile malicious nodes by leveraging the Sequential Probability Ratio Test (SPRT) [72]. 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 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, with low false negative and false positive error rates in sections 4.4 and 4.5. 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. Moreover, we evaluate our scheme through simulation in section 4.6. Our results show that our scheme detects mobile malicious nodes with only a few samples and very low false positive and negative rates. Finally, we summarize the chapter in section 4.7.

## 4.2 Assumptions

In this section, we describe the network and attacker assumptions for mobile malicious node detection scheme.

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 [9, 17, 35, 36, 58, 88]. 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.

We also 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. 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 [34], 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 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 [9, 20, 41]. The network can use one of these schemes to detect malicious nodes that virtually move with strong signal power.

### 4.3 Protocol Description

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

Our main intuition 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) [72]. The SPRT has the advantage that it reaches a decision with few samples while achieving low false positive and false negative rates [72]. We can use this 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.

Before deployment, the network operator assigns a unique ID to every sensor node. He also pre-loads secret keying materials onto each sensor node that are required for pairwise key establishment; we can use any key pre-distribution techniques for sensor networks such as [17, 88]. 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. Since secrecy of the size of each time slot is useful for our scheme, the size of each time slot is chosen randomly.

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  $i$ th time slot,  $i \geq 1$ . Let  $G_i$  be a Bernoulli random variable that indicates whether  $v$  was silent during the  $i$ th time slot;  $G_i$  is defined as:

$$G_i = \begin{cases} 1 & \text{if } S_i = 0 \\ 0 & \text{if } S_i > 0 \end{cases} \quad (4.1)$$

The success probability  $\varphi$  of a Bernoulli distribution is defined as  $\varphi = \Pr(G_i = 1) = 1 - \Pr(G_i = 0)$ . If  $\varphi$  is less than or equal to a preset threshold  $\varphi'$ , 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  $\alpha'$  and false negative rate  $\beta'$  in such a way that the false positive and false negative rates are respectively bounded by  $\alpha'$  and  $\beta'$ .

Now we present 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:

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

Assume that  $G_i$  values are independent and identically distributed.

Then  $L_n$  can be rewritten as

$$L_n = \ln \frac{\prod_{i=1}^n \Pr(G_i | H_1)}{\prod_{i=1}^n \Pr(G_i | H_0)} = \sum_{i=1}^n \ln \frac{\Pr(G_i | H_1)}{\Pr(G_i | H_0)} \quad (4.2)$$

Let  $\psi_n$  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  $\varphi_0$  and  $\varphi_1$  is as follows.  $\varphi_0$  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.  $\varphi_1$  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  $v$ .

#### 4.4 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.

##### 4.4.1 Detection Accuracy

In the SPRT, the false positive rate  $\alpha$  and false negative rate  $\beta$  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  $\alpha$  and  $\beta$  are computed as  $\alpha \leq \frac{\alpha'}{1-\beta'}$  and  $\beta \leq \frac{\beta'}{1-\alpha'}$  in [72]. The upper bound of the summation of  $\alpha$  and  $\beta$  is given by  $\alpha + \beta \leq \alpha' + \beta'$  [72]. Because  $\beta$  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.

#### 4.4.2 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  $u_1, u_2, \dots, u_k$  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  $u_2, u_3, \dots, u_k$ .

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

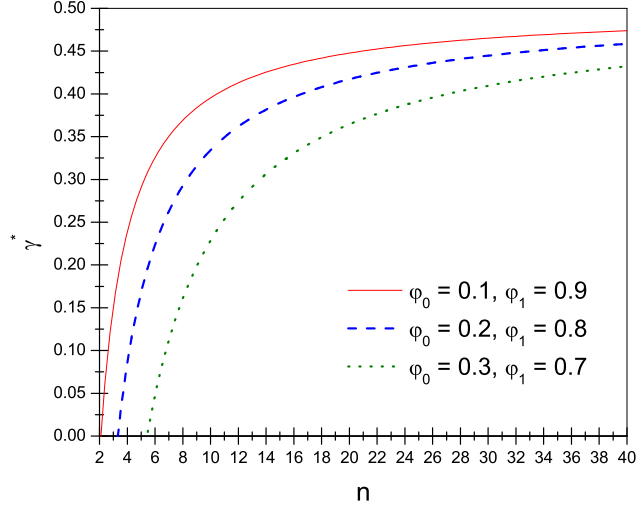


Figure 4.1: Minimum fraction of neighbor contact time slots to avoid detection ( $\gamma^*$ ) vs. number of time slots ( $n$ ).

scheme greatly limits the number of regions in which the malicious node can operate effectively.

**Lemma 1** *Let us consider the  $n$  time slots of node  $u_i$ ,  $1 \leq i \leq k$ . Let us denote by  $\gamma$  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  $\gamma n$  time slots, the number of packets sent to  $u_i$  by  $v$  is measured to be greater than zero for the first  $\gamma n$  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  $j$ th time slot corresponds to the  $j$ th sample in the SPRT. Since  $v$  leaves from  $u_i$  and moves to  $u_{i+1}$  after the first  $\gamma n$  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  $\gamma^*$  when  $\alpha' = \beta' = 0.01$ . For this study, we consider three settings of the lower and upper threshold values  $\varphi_0$  and  $\varphi_1$ :  $\varphi_0 = 0.1$  and  $\varphi_1 = 0.9$ ,  $\varphi_0 = 0.2$  and  $\varphi_1 = 0.8$ ,  $\varphi_0 = 0.3$  and  $\varphi_1 = 0.7$ . As shown in Figure 4.1,  $\gamma^*$  tends to increase as  $n$  rises in all three cases. However, the growth of  $\gamma^*$  slows such that the value of  $\gamma^*$  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 immobile malicious node detection [10, 30, 65, 68, 69, 82]. Essentially, the attacker is stuck between using mobile nodes and being detected by our scheme and using static nodes and getting detected through existing approaches.

**Lemma 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.

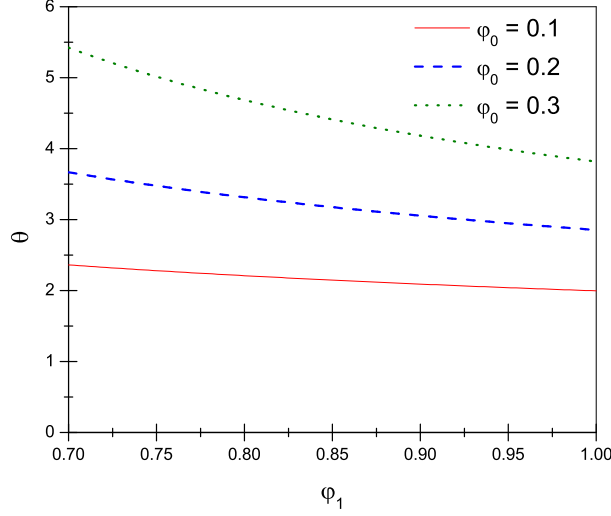


Figure 4.2: Maximum number of time slots waiting for the first neighbor contact ( $\theta$ ) vs. the upper threshold ( $\varphi_1$ ).

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  $n' \geq \frac{\ln \frac{1-\beta'}{\frac{\alpha'}{\varphi_1}}}{\ln \frac{\alpha'}{\varphi_0}}$  holds.

We now investigate how upper threshold  $\varphi_1$  affects  $\theta$  when  $\alpha' = \beta' = 0.01$ , and the lower threshold  $\varphi_0 = 0.1, 0.2, 0.3$ . As shown in Figure 4.2,  $\theta$  decreases as  $\varphi_1$  increases when  $\varphi_0$  is fixed to 0.1, 0.2, and 0.3. On the other hand,  $\theta$  increases as  $\varphi_0$  rises when  $\varphi_1$  is fixed. We see from this that small values of  $\varphi_0$  and large values of  $\varphi_1$  lead to small values of  $\theta$ . We also observe that  $\theta$  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 1 and Lemma 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  $u_{i+1}, \dots, u_k$  according to Lemma 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 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  $u_1, u_2, \dots, u_c$  and has a mobile malicious node sequentially contact the nodes in the circular list starting from  $u_1$  while sending attack packets through the contacted  $u_i$ , ( $1 \leq i \leq c$ ). 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 mobile malicious node periodically communicate with  $u_i$ . However, each sensor node randomly chooses the size of each time slot and thus it will be difficult for a mobile malicious node to correctly predict the sizes of a series of time slots of each node  $u_i$ . Accordingly, the mobile malicious node has difficulty in deciding how long it needs to stay as neighbor of each node  $u_i$  in order not to be detected by all  $c$  nodes in the circular list. This leads to an increase in the number of time slots during which each node  $u_i$  decides mobile malicious node to be absent, contributing to the detection of it. Also, 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 immobile malicious node detection [10, 30, 65, 68, 69, 82, 89]. Hence, attacker's benefits from circular movement attack are substantially limited by random selection strategy of time slot size or the existing static malicious node detection schemes.

#### 4.4.3 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  $\alpha$  at or below  $\frac{\alpha'}{1-\beta'}$ . Setting the

user-configured false positive  $\alpha'$  and user-configured false negative  $\beta'$  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 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.

#### 4.5 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 [72],  $E[n]$  is calculated as 
$$E[n] = \frac{E[L_n]}{E\left[\ln \frac{\Pr(G_i|H_1)}{\Pr(G_i|H_0)}\right]}.$$



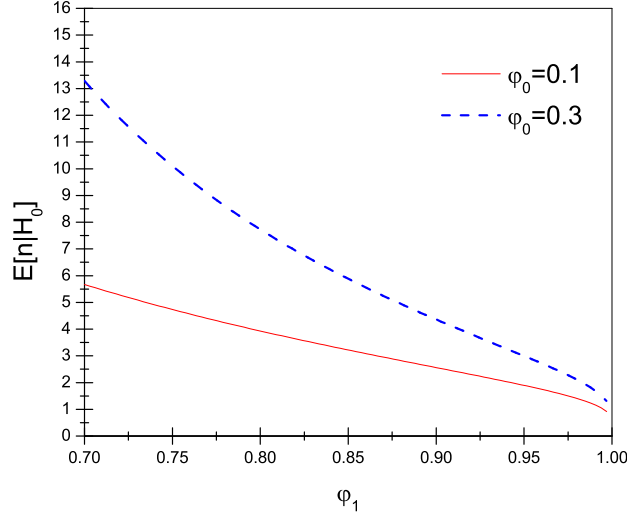


Figure 4.3: Ave. time slots to accept  $H_0$  ( $E[n|H_0]$ ) vs. the upper threshold ( $\varphi_1$ ).

We then compute the expected values of  $n$  conditioned on hypotheses  $H_0$  and  $H_1$  as follows:

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

$$E[n|H_1] = \frac{\beta' \ln \frac{\beta'}{1 - \alpha'} + (1 - \beta') \ln \frac{1 - \beta'}{\alpha'}}{\varphi_1 \ln \frac{\varphi_1}{\varphi_0} + (1 - \varphi_1) \ln \frac{1 - \varphi_1}{1 - \varphi_0}} \quad (4.3)$$

Figures 4.3 and 4.4 show how lower threshold  $\varphi_0$  and upper threshold  $\varphi_1$  affect  $E[n|H_0]$  and  $E[n|H_1]$ , respectively. Specifically,  $E[n|H_1]$  decreases as  $\varphi_1$  rises when  $\varphi_0$  is fixed to 0.1 and 0.3. This means that a large value of  $\varphi_1$  contributes to detection of mobile malicious nodes quickly with few reports. For a given value of  $\varphi_1$ ,  $E[n|H_1]$  is less when  $\varphi_0 = 0.1$  than when  $\varphi_0 = 0.3$ . This means that small values of  $\varphi_0$  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  $\varphi_0$  and a large value of  $\varphi_1$  also lead to a decrease in the number of samples required to correctly decide that an immobile benign node is not moving.

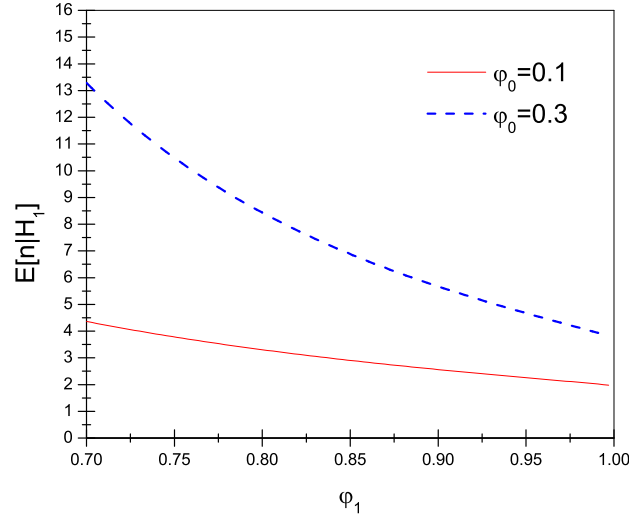


Figure 4.4: Ave. time slots to accept  $H_1$  ( $E[n|H_1]$ ) vs. the upper threshold ( $\varphi_1$ ).

## 4.6 Experimental Study

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

### 4.6.1 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,  $u$  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  $\tau$  the time period during which  $w$  stays within the vicinity of  $u$  and define a time slot as a unit of  $\tau$ .

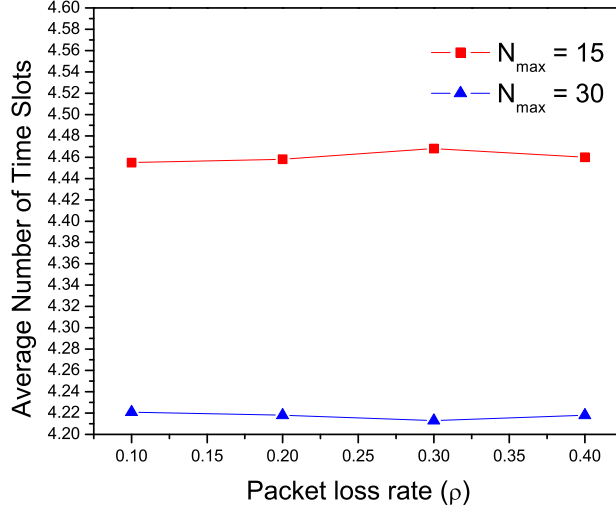


Figure 4.5: Average number of time slots vs. Packet loss rate ( $\rho$ ) in the trueNegative case.

We also set the both the user-configured false positive threshold  $\alpha'$  and the false negative threshold  $\beta'$  to 0.01, and we set lower and upper thresholds  $\varphi_0$  and  $\varphi_1$  to 0.1 and 0.9, respectively. We discuss the rationale behind these configurations in Section 4.5. To emulate packet loss due to packet collision, we modify the number of packets generated by a node with packet loss rate  $\rho$ . Specifically, we take the number of packets generated by a node ( $\eta$ ), and generate  $\eta'$  selected uniformly at random from the range  $[\eta - \eta\rho, \eta + \eta\rho]$ . This configuration reflects that the number of lost packets due to packet collision increases with greater values of  $\rho$ .

#### 4.6.2 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.

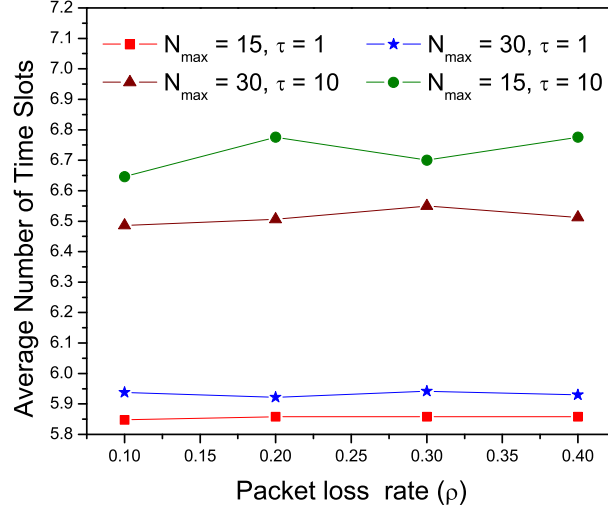


Figure 4.6: Average number of time slots vs. Packet loss rate ( $\rho$ ) in the truePositive case.

- *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  $\rho$ , and the highest false positives rate was measured as 0.01 when  $\rho = 0.3, 0.4$  and  $N_{\max} = 15$ . Thus, mobile malicious nodes were always detected and immobile benign nodes were misidentified as mobile malicious nodes with at most probability of 0.01. 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.

Second, the average number of time slots to make a decision is shown in Figures 4.5 and 4.6. We present the results for two cases. In the *trueNegative* case

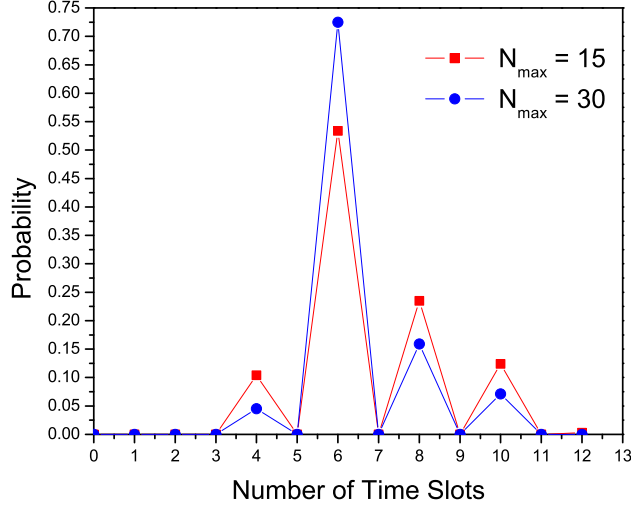


Figure 4.7: Probability distribution of the number of time slots when  $\rho = 0.4$  and  $\tau = 10$ .

(Figure 4.5), node  $u$  decides correctly that an immobile benign node  $v$  is not a mobile malicious node. In the *truePositive* case (Figure 4.6),  $u$  correctly decides that mobile malicious node  $w$  has moved and should be blocked.

In the *trueNegative* case, the average number of time slots reaches its maximum of 4.47 when  $\rho = 0.3$  and  $N_{max} = 15$ . In the *truePositive* case, the average number of time slots reaches its maximum of 6.78 when  $\rho = 0.2, 0.4$ ,  $N_{max} = 15$ , and  $\tau = 10$ . Thus,  $u$  reaches correct decisions in a few time slots in both cases. Moreover, we observe that the average number of time slots has a near-constant value, irrespective of the values of packet loss rate  $\rho$  in both cases. This indicates that an increase in  $\rho$  rarely affects the average number of time slots as long as not all packets are lost.

In both cases, the average number of time slots when  $N_{max} = 30$  is less than when  $N_{max} = 15$ . This indicates that for larger  $N_{max}$ , the chance is greater that no packets are generated by  $v$  or  $w$ , leading to a rise in the average number of time slots. In the *truePositive* case, the average number of time slots when  $\tau = 1$  is less than one when  $\tau = 10$ . This is because an increase in  $\tau$  leads to a rise in the time period

during which mobile malicious node  $w$  stays within the vicinity of  $u$  and thus delays moving the test toward  $H_1$ .

Finally, Figure 4.7 shows the probability distribution of the number of time slots in the case of truePositive when  $\rho = 0.4$  and  $\tau = 10$ . For this distribution, we examine two scenarios:  $N_{max} = 15$  and  $N_{max} = 30$ . A total of 63.8% and 77% of the cases fall in the range from 4 to 6 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 six time slots in most cases.

#### 4.7 Summary

In this chapter, 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 its 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.

## CHAPTER 5

### STATIC REPLICA NODE DETECTION

In Chapters 3 and 4, we proposed static and mobile node compromise detection schemes. Even though these schemes achieve efficient and robust detection and revocation of compromised nodes, it cannot be applied to directly prevent widespread node compromise by replica node attacks. To mitigate this limitation, we propose distributed replica node detection schemes in static sensor networks. The proposed schemes are based on the realistic assumption that nodes are deployed in groups, which is realistic for many deployment scenarios. By taking advantage of group deployment knowledge, the proposed schemes perform replica detection in a distributed, efficient, and secure manner. We show analytically and in simulation that the proposed schemes achieve effective and robust replica detection capability with substantially lower communication, computational, and storage overheads than prior work.

#### 5.1 Overview

In this section, we present an overview of the static replica detection schemes.

Parno et al. [58] first proposed replica detection schemes in wireless sensor networks. The main idea of these schemes is to have nodes report *location claims* that identify their positions and attempt to detect conflicting reports that signal one node in multiple locations. This requires every node to sign and send a location claim, and verify and store the signed location claim of every other node. Thus, this requirement

results in significant communication, computation, and storage overheads, which may cause the sensor nodes to run out of power.

In this work, we seek ways to achieve effective and robust replica node detection capability with lower communication, computation, and storage overheads than prior work. To meet this goal, we propose several replica detection schemes based on the assumption that sensor nodes will be deployed in groups and that all nodes know approximately where each group will be deployed. This should be realistic for many deployment scenarios, as sensor nodes can be dropped from airplanes or scattered over an area by hand. In such a case, the sensors can be pre-loaded with relevant knowledge about their own group’s membership and all group locations. Then, the sensors in the same group should be deployed at the same time in the location given to that group.

Our schemes adapt the location claim idea from [58]. However, given that nodes are deployed in groups, our schemes allow most nodes to communicate without generating any location claims as long as they are able to directly send messages to at least one of their group members. This simple idea allows us to significantly reduce the overhead of sending, receiving, and verifying location claims. Additionally, if we assume loose time synchronization for the sensors, we can allow nodes to accept messages from any node that has been deployed within a small window of its expected deployment time. This assumption allows us to eliminate most of the overheads of our approach, except when nodes are not deployed close to their expected deployment time nor close to their expected deployment location.

We validate the effectiveness and efficiency of the proposed schemes through analysis and simulation. In particular, we demonstrate that our schemes are very effective at detecting and blocking replica nodes while incurring substantially lower overheads than those in [58]. Our schemes can be made arbitrarily efficient by increas-



ing the accuracy of deployment knowledge. Networks with very accurate deployments, in both time and location, can reach practically zero overhead for benign nodes.

The rest of chapter is organized as follows. Section 5.2 presents the underlying assumptions for static replica detection schemes. Section 5.3 proposes the basic approach of group-based replica detection schemes. Section 5.4 proposes a location claim approach to mitigate the limitations of the basic approach. Section 5.5 proposes a multi-group approach to enhance the security resilience and performance of the location claim approach. Section 5.6 proposes a deployment time check technique for further enhancements of the Schemes 5.4 and 5.5. Section 5.7 compares our schemes to [58] in terms of the communication, computational, and storage overheads. We summarize the chapter in Section 5.8.

## 5.2 Assumptions

In this section, we describe the network and attacker assumptions for the static replica detection schemes.

We study replica detection in a two-dimensional *static* sensor network where the locations of sensor nodes do not change after deployment. We also assume that all direct communication links between sensor nodes are bidirectional, which is common in the current generation of sensor networks.

We assume that an adversary may compromise and fully control a subset of the sensor nodes. However, we place some limits on the ability of the adversary to compromise nodes. If the adversary can compromise a major fraction of the network, he will not need or benefit much from the deployment of replicas. We assume that the attacker can identify and compromise a substantial fraction of the nodes in a small area. He will subsequently make replicas of one or more of these nodes and attempt to distribute them throughout the network. We do not specifically limit

Table 5.1: Frequently used notations.

$N$	total number of sensor nodes.
$g$	total number of groups.
$m$	group size.
$b$	average neighbor size of a node.
$G_u$	group to which node $u$ belongs.
$  $	concatenation symbol.

the attacker from compromising nodes throughout the network, but we note that there is little benefit to the attacker of having a replica node in the same area as another compromised node. The compromised node can just as easily report fake data, participate in local control protocols, and eavesdrop on messages sent through it. Furthermore, if the attacker needs one compromised node to accompany each replica node in the network, there will be a very high cost for launching replica node attacks. This assumption goes unstated but is implied by the use of signed location claims in other replica detection schemes [13, 58]. It is also worth noting that the attacker does not benefit from having multiple replicas of a single node in the same small region. For example, in a false data injection attack, it would be easy to ensure that only one of the replica nodes' data values at a time is accepted by the data aggregators. In local control protocols, each node could only have input once, so multiple nodes with the same ID would not have more influence in a region than a single node.

### 5.3 Scheme I: The Basic Approach

In this section, we present the details of our techniques to stop node replica attacks with group deployment knowledge in a two-dimensional static sensor network, where the locations of sensor nodes do not change after deployment. The key as-

sumption in our work is the use of a group deployment strategy that was described in Chapter 2. Given this pre-deployment knowledge, it is still non-trivial to perform replica detection in sensor networks due to the difficulty in *precisely* deploying sensor nodes at their expected group deployment points. Hence, we may make wrong decisions when pre-deployment knowledge is used for replica detection and there are errors during the actual deployment. How the system handles these deployment errors will be an important issue explored in this chapter. We start with a basic approach and then show how to improve the security as well as the performance of this approach with additional knowledge about the deployment of sensor nodes. Table 5.1 lists notations used frequently in our discussion.

As mentioned above, our schemes assume that sensor nodes are deployed group by group, and that each group is expected to be deployed towards a *deployment point* that can be pre-determined. Prior to deployment, the network operator loads the pre-determined deployment coordinates of every group onto every sensor node. Recall that sensor nodes in the same group are very likely to be close to each other after deployment. Our proposed schemes use this knowledge to stop node replica attacks. A simple way to do this would be to have each node only accept messages from members of its own group. However, this stops inter-group communication. We extend this idea by defining nodes that are close to their group deployment point as *trusted* and nodes that are far from their group deployment point as *untrusted*. Nodes will accept messages from trusted neighbors, while ignoring messages from untrusted nodes. Let us now describe the basic approach.

### 5.3.1 Protocol Description

Before deployment, the sensor nodes in the network are divided into  $g$  groups each with an equal number of nodes  $m$  and a unique group ID. The ID of every

sensor node has two parts: the group ID and a unique ID within the group. Keying materials are also pre-loaded onto each sensor node for pairwise key establishment; we can use any key predistribution technique for sensor networks, such as the ones in [7, 17, 50, 88]. We assume that every message in the system is protected by a pairwise key, and that all messages include the ID of the sender in plaintext. This prevents simple spoofing attacks, as receivers will use the pairwise key matching the ID seen by all intermediate nodes.

Suppose that a sensor node  $u$  (a member of the group  $G_u$ ) receives a request from its neighbor node  $v$  (a member of the group  $G_v$ ) to forward a message. Node  $u$  first checks whether the distance between the deployment points of groups  $G_u$  and  $G_v$  is smaller than a pre-defined system-wide threshold distance,  $d$ . If so,  $u$  believes that node  $v$  is a trusted neighbor and accepts and forwards messages from node  $v$ . Otherwise, it will ignore all messages from node  $v$ . Intuitively, a node that is not near its own group members will be isolated and unable to send messages. The deployment points used in the above protocol are pre-determined before the deployment and can be easily inferred from the group IDs. Neither  $u$  nor  $v$  can claim an incorrect deployment point even if they are compromised.

From the above description, we can clearly see that the basic scheme does not introduce any significant communication, computation, or storage overhead. Each node can immediately determine whether to forward the messages from other sensor nodes by only looking at their IDs. This approach, as the others proposed in this chapter, makes these decisions just once, whenever a node first contacts its neighbors. We now examine the security and effectiveness of the basic approach.

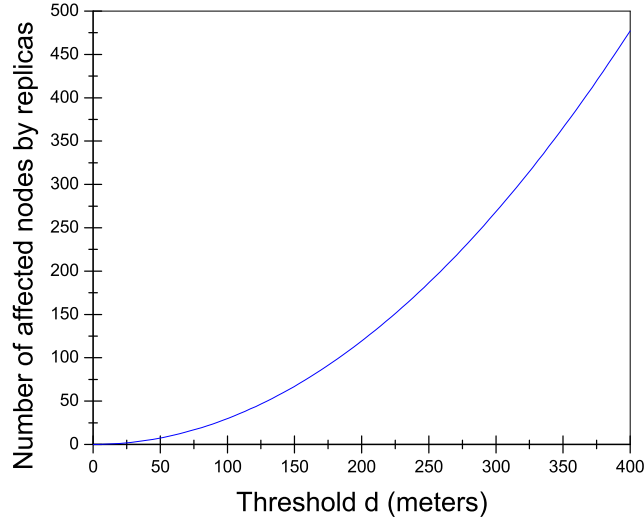


Figure 5.1: The number of nodes affected by replica nodes.

### 5.3.2 Security Analysis

Suppose that the adversary has already compromised node  $u$  and placed a number of replicas of  $u$  in the field. From our protocol description, we know that a replica node  $u'$  of  $u$  cannot change its group deployment point since it is pre-determined before deployment. Hence, we see that only the nodes that are supposed to be deployed less than  $d$  meters away from the group deployment point of node  $u$  will accept replicas with ID  $u$  as their trusted neighbor nodes. In other words, node  $u$ 's replicas will be accepted by the following set of nodes:  $O = \{i \mid \text{dist}(i, u) \leq d\}$ , where  $\text{dist}(i, j)$  is the distance between the deployment points of groups  $G_i$  and  $G_j$ .

We now determine the maximum impact of a replica node attack. Let  $m$  be the number of nodes in each group,  $g$  be the total number of groups, and  $S$  be the size of the target deployment field. Assume that the pre-determined group deployment points are randomly distributed in the field. Consider a particular group  $G_u$ . The average number of groups whose deployment points are no more than  $d$  meters away from the deployment point of group  $G_u$  can be estimated by  $\frac{(g-1) \times \pi \times d^2}{S}$ . We define

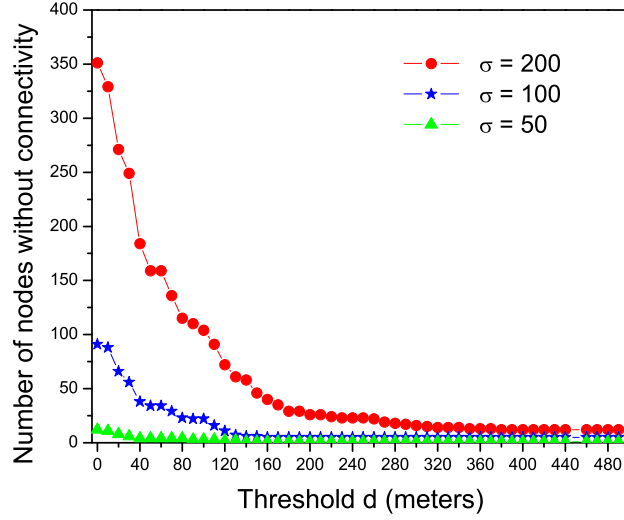


Figure 5.2: The number of nodes without connectivity.

the *affected* nodes as those nodes that will accept the replicas of a particular node  $u$  as their trusted neighbors. Thus, the average number of the affected nodes can be estimated by

$$\frac{m \times (g - 1) \times \pi \times d^2}{S}$$

To concretely show the effect of replica nodes, we will examine a specific scenario. Suppose that  $N = 1,000$  sensor nodes are deployed in a square field of  $S = 1000 \times 1000$  square meters. Each sensor node can communicate with other sensor nodes in a radius of 50 meters. These 1,000 sensor nodes are divided into  $g = 20$  groups with  $m = 50$  nodes in each group. Figure 5.1 shows the effect of replica nodes. We can see that the number of affected sensor nodes increases with  $d$ . When  $d = 81$  meters, less than 2% of the nodes are affected. However, when  $d = 185$  meters, over 10% of the nodes become affected. Thus, we need to set  $d$  as small as possible to mitigate the impact of node replica attacks on the network. However,

having a small value for  $d$  causes many nodes to be unable to communicate, as shown in the next part of our analysis.

### 5.3.3 Connectivity

In the basic approach, every sensor node will reject forwarding data packets for those untrusted neighbor nodes. However, according to our deployment model, some benign nodes may be deployed at placement points that are far from their group deployment points. As a result, these benign nodes will be considered as untrusted and rejected by their neighbors. These are honest nodes, but they are unable to communicate and are effectively dead on arrival. In an extreme case, the network could be partitioned. We now examine *the number of nodes without connectivity* in benign situations.

To investigate the number of nodes disconnected from the rest of the network, we developed a simple simulation using the same network settings as above. Figure 5.2 shows the number of disconnected nodes for different values of threshold distance ( $d$ ) and standard deviation ( $\sigma$ ). We observe that the number of such nodes decreases rapidly with increasing  $d$ . However, increasing  $d$  will also allow the attacker to generate a larger impact (more replicas being accepted as trusted neighbors) on the network. When  $\sigma = 50$ , i.e., the deployment is fairly accurate, almost all nodes have connectivity for  $d > 100$  meters. When  $\sigma = 200$  and  $d = 100$  meters, nearly 30 nodes are affected by the attacker, but 104 nodes are without a connection. When  $\sigma = 200$  and  $d = 200$  meters, the attacker impacts nearly 119 nodes while 26 benign nodes remain disconnected. Obviously, we have to make a trade-off for the choice of  $d$ , and that tradeoff is significantly impacted by the accuracy of the sensor deployment.

## 5.4 Scheme II: Location Claim Approach

In the basic approach, sensor nodes only forward messages for their trusted neighbors. As discussed, when the deployment is not very accurate, many benign nodes may be rejected by their neighbors for message forwarding, and the sensor network may be poorly-connected or even partitioned when the application requires high resilience against replica node attacks. To address this problem, we propose that sensor nodes also forward messages from untrusted neighbors as long as they provide provable *evidence* that they are not replicas. The evidence includes the location of the node requesting message forwarding. The evidence from an untrusted node will be sent to a pre-determined location for replica detection; when two conflicting pieces of evidence reach this location, the replica will be detected.

### 5.4.1 Protocol Description

In Scheme II, in addition to the keying materials for pairwise key establishment, every sensor node also gets the keying materials for generating digital signatures. We will use an identity-based public key scheme [2, 12, 63]. It has been demonstrated that public key operations can be efficiently implemented in static sensor devices [33, 49, 73]. Moreover, most existing replica detection schemes for sensor networks [13, 58, 90] employ an identity-based public key scheme. Scheme I only requires the knowledge of group deployment points of sensors. In Scheme II, we also assume that a secure software-based localization method has been employed in the network such as the ones in [6, 46, 51]. This allows a sensor node to discover its actual placement point after deployment. The scheme consists of four phases as described below.

**Phase 1: Deployment.** We define the *deployment zone* of a group as a circle centered at the group's deployment point with radius  $R_z$ . Here  $R_z$  is a configurable parameter of the system and we will set it to balance security and performance. When



discussing a given node  $u$ , we call the corresponding deployment zone of group  $G_u$  the *home zone*. After deployment, every node  $u$  discovers its real location,  $L_u$ . If  $u$  resides outside its home zone, it produces a *location claim*  $C_u = \{u||L_u||Sig_u\}$ , where  $Sig_u$  is the signature generated by  $u$ 's private key.

**Phase 2: Neighbor Discovery.** After deployment, every sensor node  $u$  discovers a set of neighbors  $N(u)$  and asks for an authenticated location claim from every node  $v \in N(u)$ . Upon receiving a claim request,  $v$  sends  $u$  a location claim  $C_v$  if  $v$  is placed outside its home zone. Otherwise, it just sends  $u$  the message  $M_v = \{v||L_v||MAC_{K_{uv}}(v||L_v)\}$ , where MAC stands for Message Authentication Code and  $K_{uv}$  is the shared secret key between  $u$  and  $v$ . Any node  $v$  will be removed from  $N(u)$  if  $C_v$  or  $M_v$  fails to authenticate. Also, if  $v$  claims a location  $L_v$  such that the distance between  $L_u$  and  $L_v$  is larger than the communication range of  $u$ , then  $v$  will be removed from  $N(u)$ . After this,  $u$  checks every node  $v \in N(u)$  to see if it is deployed in the right place (i.e., node  $v$ 's home zone). This can be done by checking whether the distance between  $L_v$  and the deployment point of group  $G_v$  is less than  $R_z$ . If node  $v$  is indeed in its home zone, then  $u$  will mark node  $v$  as a *trusted* node; otherwise, it will mark  $v$  as *untrusted*.

**Phase 3: Claim Forwarding.** A sensor node  $u$  will forward regular messages from its untrusted neighbor  $v$ , only if it has received and verified a location claim  $C_v$ . When  $u$  gets claim  $C_v$ , it forwards  $C_v$  to  $v$ 's home zone with probability  $P_f$  to ensure that at least one claim reaches the home zone of  $G_u$ . If node  $u$  is going to forward the claim, it sends  $C_v$  to the deployment point of group  $G_v$ . Since most nodes in  $G_v$  will be deployed in  $v$ 's home zone, delivering the claim to the deployment point of  $G_v$  will reach many sensor nodes in  $G_v$ .

**Phase 4: Detection and Revocation.** Once an authenticated location claim reaches the home zone of node  $v$ , it will be flooded throughout the home zone of  $G_v$ .

Such local flooding is inexpensive since it is confined to a small area. This ensures that every node in group  $G_v$  residing in  $v$ 's home zone will know  $C_v$ . If the nodes in the home zone of  $v$  receive two conflicting claims, they conclude that node  $v$  has been replicated. The two conflicting claims can be used as evidence to revoke node  $v$  from the network. This can be most easily accomplished by broadcasting the conflicting claims throughout the network.

We can employ either a coordinated or uncoordinated approach to broadcast the conflicting claims. In the coordinated approach, each node takes a role of broadcasting the conflicting claims in a round robin manner. In the uncoordinated approach, the node that broadcasts the conflicting claims is chosen in accordance with a pseudo-random number generator based on the ID value of the replica. Upon receiving the conflicting claims, every node can independently verify them and take revocation on node  $v$ . Note that revocation can be lazy, in that a node does not need to verify the claims until  $v$  attempts to be its neighbor. In this way, replicas are detected and revoked in a fully distributed manner.

#### 5.4.2 Security Analysis

Suppose node  $u$  has been compromised. If all the replicas of  $u$  generate the same location claim  $C_u$ , they can only affect the nodes that are no more than  $r$  meters away from  $L_u$ , where  $r$  is the communication range of a node. This will not offer the adversary much benefit. In addition, if all the claims include locations that reside in the home zone of node  $u$ , they can only affect the nodes that are no more than  $2r$  meters away from the deployment point of group  $G_u$ . This also does not give the attacker much benefit since it can only impact a small area. As a result, we focus on the probability  $P_d(c)$  of detecting replica nodes when the attacker makes  $c$  different location claims outside of the home zone of node  $u$ .

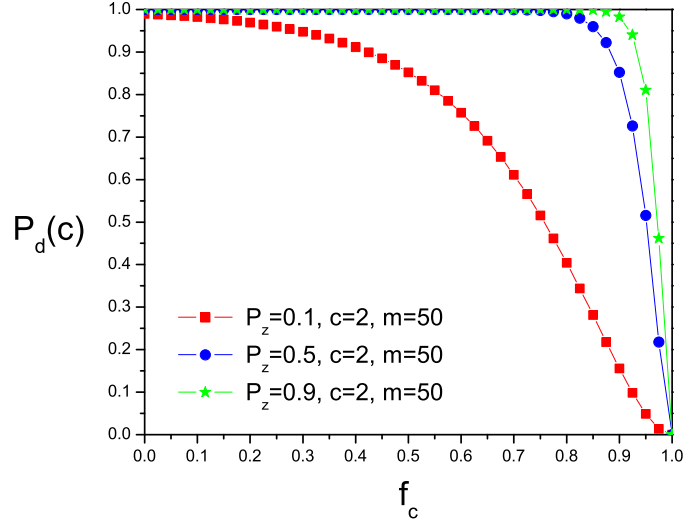


Figure 5.3: The detection probability  $P_d(c)$  as a function of the fraction of compromised nodes  $f_c$ .

Note that, for the calculation of  $P_d(c)$ , we ignore the possibility of compromised nodes failing to forward the location claim. If any significant fraction of the neighbors of a replica node are compromised, they already have substantial influence over that part of the network and hence adding replicas does not substantially help them, as discussed in Section 5.2.

For a particular location claim of node  $u$ , the probability  $P_z$  that a given node of group  $G_u$  resides in the home zone of  $u$  can be estimated by

$$\begin{aligned}
 P_z &= \int_0^{2\pi} \int_0^{R_z} f(\rho \cos \theta, \rho \sin \theta) \rho d\rho d\theta \\
 &= 1 - e^{-\frac{R_z^2}{2\sigma^2}}
 \end{aligned} \tag{5.1}$$

where  $\sigma$  is the standard deviation of the two-dimensional Gaussian distribution, and  $R_z$  is the radius of the home zone of node  $u$ .

Since these  $c$  claims are for locations that are outside of the home zone of node  $u$ , they will all be delivered to  $u$ 's home zone. Assume that every claim can be reliably

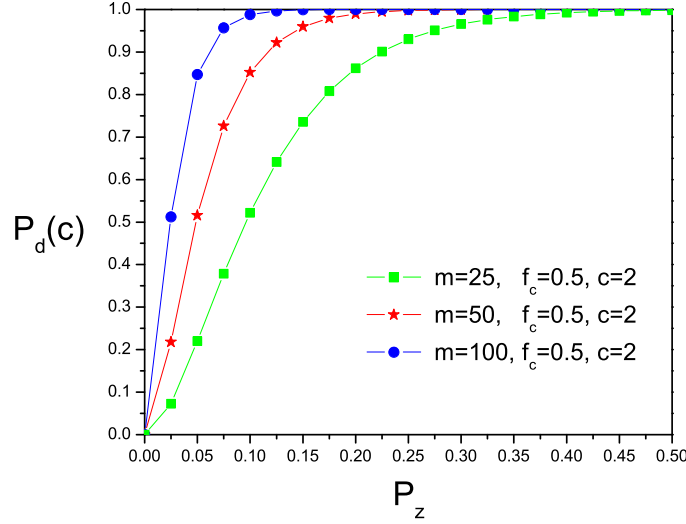


Figure 5.4: The detection probability  $P_d(c)$  as a function of the group deployment accuracy  $P_z$ .

delivered to the home zone of  $u$ . Let  $m$  and  $f_c$  be the group size and the fraction of compromised nodes in the network. We estimate the probability  $P'$  that location claim can find at least one benign node in the home zone of node  $u$ . The probability  $P'$  can be estimated as

$$P' = 1 - [1 - P_z \times (1 - f_c)]^m$$

With  $P'$ , we can estimate the probability  $P_d(c)$  as:

$$P_d(c) = 1 - (1 - P')^c - c \times P'(1 - P')^{c-1} \quad (5.2)$$

Using Equation 5.2, we study how  $P_d(c)$  is affected by the different values of  $f_c$  and  $P_z$ . As shown in Figure 5.3, we see that Scheme II achieves a detection probability of  $P_d(c) = 0.99$  as long as  $P_z \geq 0.5$ , even if 80% of the nodes in the home zone are compromised. If  $P_z$  reaches a very low value such as 0.1, and there is a high fraction

of compromised nodes, the detection capability would be degraded. However, this scenario implies that the accuracy of group deployment was very poor.

We also study how  $m$  and  $P_z$  impact  $P_d(c)$ . From Figure 5.4, we observe that the replica detection probability tends to increase in proportion to the rise of group size ( $m$ ) when  $P_z \leq 0.4$ . Moreover, Scheme II achieves a detection probability of  $P_d(c) = 0.99$  as long as  $m \geq 50$  for  $P_z \leq 0.2$  and  $f_c = 0.5$ . This means that the group size of  $m = 50$  is enough to achieve replica detection with very high probability of 0.99 even when the group deployment accuracy is low and there is a substantial fraction of compromised nodes.

Since every location claim is digitally signed, an adversary cannot make uncompromised nodes appear to be replicas by faking their location claims. Thus, the replica detection scheme works without any false positives.

#### 5.4.3 Communication Overhead

We define communication overhead as the average number of location claims that are sent and received by nodes in the network. First, we emphasize that claims are only generated and sent once at the time when new untrusted nodes are deployed. Thus, Scheme II has lower overheads in the lifetime of the network than schemes with repeated claim generation and verification, such as in [13].

We begin by estimating the communication overhead in benign situations when there are no replica nodes in the field. In this case, the sensor nodes that reside in their home zone will not incur any additional overhead. Thus, on average, the fraction of sensor nodes whose location claims need to be forwarded to their home zones can be estimated by  $1 - P_z$ . This means that we often prefer a large probability  $P_z$  of being in the home zone for communication efficiency. This is also beneficial to replica detection, as shown above, so the accuracy of deployment is very important to the

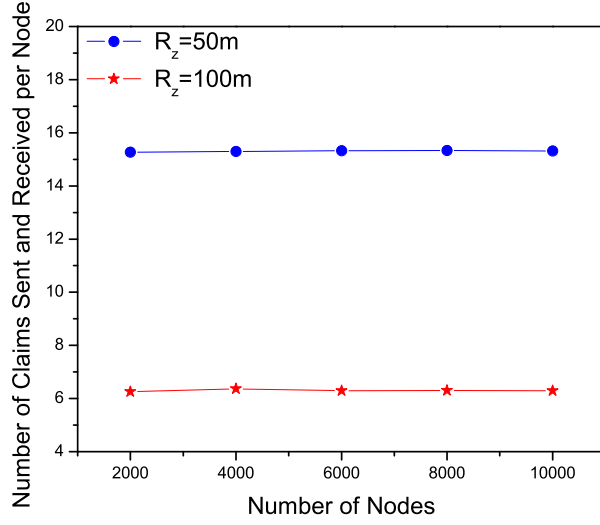


Figure 5.5: Communication overhead per node.

scheme. If deployment accuracy is poor, we can keep  $P_z$  high by increasing  $R_z$ . The downside is that it will increase the size of home zones within which replica nodes would be trusted.

In benign situations, the distance between the node and its home zone will be relatively small. Just as we defined home zones, we can also define *communication bands* around the home zone. Each communication band  $B_i$  is a circle around the group deployment point with radius  $R_i = R_{i-1} + \alpha r$ , where  $r$  is the communication radius of each node and  $\alpha$  is a fraction set to ensure that every node in  $B_{i+1}$  can send a packet to at least one node in  $B_i$ . Band  $B_0$  has radius  $R_z$  and is the same as the home zone. Inductively, this means that  $R_i = R_z + i \times \alpha r$ .

The probability that a node exists in a given band  $B_i$  is given by:

$$P_i = \left(1 - e^{-\frac{R_i^2}{2\sigma^2}}\right) - \left(1 - e^{-\frac{R_{i-1}^2}{2\sigma^2}}\right)$$

We can conservatively assume that if a node is in band  $B_i$ , then all of its neighbors further from the deployment point are in band  $B_{i+1}$ . Thus the worst case communication overhead per node can be estimated as:

$$C_s = 2bP_f \sum_{i=1}^{\infty} P_i(i+2)$$

where  $b$  is the average neighbor size.

We developed a simple simulation experiment to investigate the average case of communication overhead per node. We simulated the scheme by varying the number of nodes from 2000 to 10000 in a field of  $500 \times 500$  square meters. In this simulation, we set  $\sigma = 50$ , and let  $m = 50$  be the number of nodes in a group and  $r = 50$  meters be the communication radius of each node. We also configured  $P_f$  in such a way that one claim per node is forwarded to the node's home zone. Figure 5.5 shows the communication overhead per node when  $R_z = 50, 100$  meters. We observe that the communication overhead is quite reasonable, less than 15.33 and 6.37 operations per node in all cases when  $R_z = 50$  meters and  $R_z = 100$  meters, respectively.

#### 5.4.4 Computation and Claim Storage Overhead

We define computation and claim storage overhead as the average number of public key signing and verification operations per node and the average number of claims that need to be stored by a node, respectively. A sensor node residing outside its home zone only needs to perform a single signature generation operation on its location. Assume that every sensor node has  $b$  neighbors on an average and  $m$  is the group size. Each of these neighbors is outside of its home zone with probability  $(1 - P_z)$ . Thus, if a sensor node resides outside of its home zone, it needs to verify no more than  $b(1 - P_z)$  signatures on average. When this node resides in its home zone,

it also needs to verify and store those signatures forwarded to the home zone. Note that it only needs to verify and store up to two different signatures for the same node. The number of signatures to be stored and verified for the location claims forwarded to the home zone is bounded by  $2 \times (1 - P_z) \times m$ , even if there are replica nodes. Hence the average number of signatures a sensor node needs to store and verify is at most  $(1 - P_z)2m$  and  $(1 - P_z)(b + 2m)$ , respectively. We can see concretely how much overhead this incurs by studying a sample scenario. Using Equation 5.1 and setting  $R_z = 2\sigma$ , we get  $P_z = 0.865$ . Then for  $b = 40$  and  $m = 50$ , each node needs to store 13.5 claims and perform 18.9 signature verifications.

#### 5.4.5 Risk of Denial of Service

We note that the attacker could attempt a denial of service (DoS) attack by flooding fake claims throughout the system. This would cause increased computational and communication overhead and could also cause sensor nodes to run out of power. This is a risk in all replica detection schemes, including those in [13, 58]. The following attributes of our design keep this from being an unacceptable risk:

- The communication costs of flooding could be generated via almost any protocol, including those for data gathering, routing, and control. Our protocol introduces no new vulnerabilities in this regard.
- The computational costs of checking flooded signatures would be borne by only one node per signature: either the neighbor who verifies the claim or the receiver in the home zone if the claim is initiated through a compromised node.
- The number of claims initiated by any single node is limited. Each neighbor of a given node  $v$  will only verify and forward the first claim that  $v$  sends it, since that is all that is required for our scheme.



Thus, Scheme II introduces a limited additional risk of DoS, in line with the risks introduced by prior schemes. Preventing these risks completely would be challenging, and we leave it to future work.

### 5.5 Scheme III: Multi-Group Approach

Scheme II is very effective at detecting replicas with low communication and computational overheads. It also achieves high replica detection capability even if attacker compromises 80% of the nodes in a given home zone. However, the scheme could become ineffective for a more powerful attacker. In particular, if the attacker compromises or physically destroys all the nodes in group  $G_u$ 's home zone, then none of the location claims for nodes in  $G_u$  will reach their destination. Alternatively, the attacker could populate the home zone of node  $v$ ,  $G_v$ , with many replicas of  $v$  that act as black holes for location claims from other replicas of  $v$ .

To protect against this kind of aggressive adversary, we introduce Scheme III. In this scheme, every sensor node sends its neighbor's location claims to multiple groups rather than a single group. This greatly improves our scheme's robustness, as tremendous effort would be required by the attacker to undermine all the groups used in detection. While this scheme has higher communication overhead, it can provide a trade-off between the overhead and resilience to attack.

#### 5.5.1 Protocol Description

The protocol is similar to Scheme II. The difference is how we select groups to which location claims are sent. Specifically, a neighbor  $u$  of an untrusted node  $v$  will select multiple groups, called *detector groups* and forward  $v$ 's location claims to each of the detector groups with probability  $P_f$ . Additionally,  $u$  will forward the claim

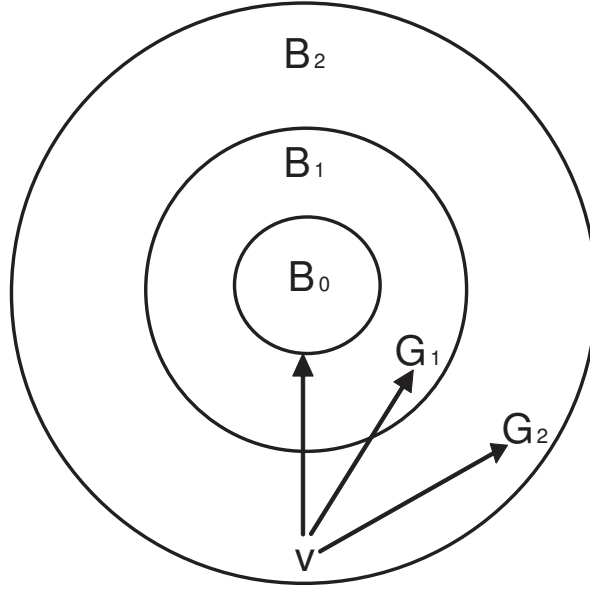


Figure 5.6: Multi-group claim forwarding.

to  $v$ 's group,  $G_v$ . Any of these groups that receives conflicting claims will initiate revocation on the replica node, thus creating greater resilience to node compromise.

We keep overhead low by selecting detector groups close to the home zone of the untrusted node. Specifically, node  $u$  selects the detector groups by using the communication bands of group  $G_v$ . Assume that  $v$  is placed in band  $B_i$ , where  $B_0$  is  $v$ 's home zone. For each band  $B_j$  between  $B_i$  through  $B_1$ , node  $u$  will select a detector group  $G_j$  from among the groups that have a deployment point in  $B_j$ . Node  $u$  can select the group by using a pseudo-random number generator based on the values of  $G_v$  and  $j$ , e.g. taking a cryptographic hash of  $(G_v || j)$ . For some band  $B_j$ , there may be no group with a deployment point in  $B_j$ . In this case,  $u$  must identify the next closest communication band with at least one group deployment point, say band  $B_l$ , and select the group pseudorandomly from that band.

For example, consider the communication bands of group  $G_v$ . Let us denote the detector groups for bands  $B_1$  and  $B_2$  by  $G_1$  and  $G_2$ , respectively. As shown in Figure 5.6, node  $v$ 's claims are forwarded to the home zones of groups  $G_1$  and  $G_2$  as well as its home zone.

### 5.5.2 Security Analysis

Similar to Scheme II, we can configure the system parameters of Scheme III to ensure detection with very high probability. However, the main benefit of Scheme III is that the attacker needs to compromise multiple groups of nodes to prevent replicas from being detected. Let us assume that the attacker employs  $k$  replicas of node  $u$  in the network. Moreover, assume that  $k$  replicas are placed in bands  $B_{\tau_1}, B_{\tau_2}, \dots, B_{\tau_k}$ , where  $0 < \tau_1 \leq \tau_2 \leq \dots \leq \tau_k$ . Note that  $\tau_k$  detector groups are selected for band  $B_{\tau_k}$ . Also,  $u$ 's group serves as a virtual detector group. Hence, at least two conflicting claims of  $u$  will be forwarded to the home zones of  $\tau_{k-1}$  detector groups as well as  $u$ 's home zone. Accordingly, the attacker needs to compromise  $mP_z(\tau_{k-1} + 1)$  nodes in order to undermine the scheme. This means that the farther away the attacker places replicas from their home zone, the more nodes he needs to compromise to avoid replica detection.

### 5.5.3 Communication Overhead

Obviously, Scheme III requires more overheads than Scheme II, since a claim is forwarded to multiple home zones. However, if the group deployment accuracy is good, most nodes will be classified as trusted and even untrusted nodes will be placed closer to band  $B_0$ . Hence, Scheme III will work with relatively small additional overheads in a benign situation when compared with Scheme II. On the contrary, if the attacker places multiple replicas as far away from each other as possible, substantial overheads will be incurred by employing Scheme III. We can further limit these overheads by extending the idea of Scheme I and blocking communications from nodes that are very far from their group deployment point. Note that, unlike in Scheme I, we can choose a much larger value for the allowed distance  $d$  that a node can be from its deployment point. The value of  $d$  can be set to ensure that no legitimate

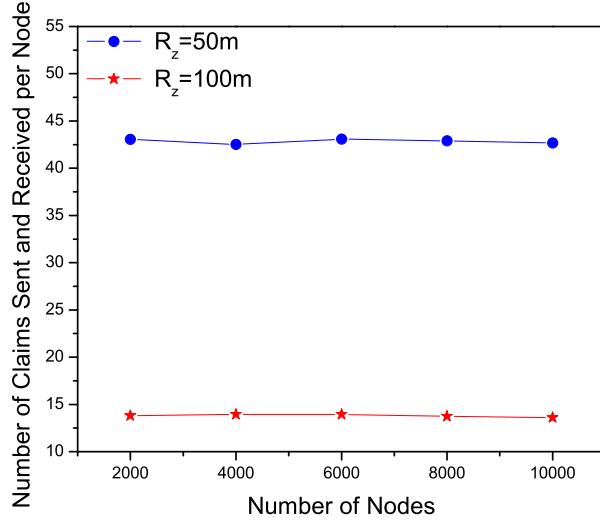


Figure 5.7: Communication overhead per node.

nodes are blocked given reasonable deployment error, while still limiting overheads from untrusted nodes that are very far from their deployment points.

We now provide an analysis of the communication overhead. Assume that a node  $v$  is placed in the band  $B_i$ . Node  $v$  sends claims to the home zones of  $i$  detector groups as well as its home zone. Thus, the worst case communication overhead per node  $C_m$  is given by:

$$C_m = C_s + 2bP_f \sum_{i=1}^{\infty} P_i \sum_{j=1}^i (\beta_j + 1)$$

where  $C_s$  is the worst case communication overhead per node in Scheme II and  $\beta_j$  is the maximum number of bands between node  $v$  and the home zone of detector group selected for band  $B_j$ .

To investigate the average case of communication overhead per node, we extended the simulation environment used for studying Scheme II. Figure 5.7 shows the communication overhead per node when  $R_z = 50, 100$  meters. We observe that the communication overhead is less than 43.1 and 14.0 operations per node in all cases

when  $R_z = 50$  meters and  $R_z = 100$  meters, respectively. The communication overhead of Scheme III is approximately three and two times as much as that of Scheme II when  $R_z = 50$  meters and  $R_z = 100$  meters, respectively. However, greater security resilience is achieved than Scheme II at the cost of this additional overhead. Thus, the communication overhead of Scheme III is still reasonable as a trade-off for greater security resilience.

#### 5.5.4 Computation and Claim Storage Overhead

Similar to Scheme II, a sensor node residing outside its home zone only needs to perform a single signature generation operation on its location. Each node's claims are forwarded to multiple detector groups' home zones as well as its home zone. The average number of signatures that a sensor needs to store and verify is at most  $2m(1 - P_z)$  in Scheme II. Thus, the average case of storage and signature verification overhead per node is bound by  $2m(1 - P_z)(1 + D_{max})$  and  $(1 - P_z)(b + 2m(1 + D_{max}))$ , respectively, where  $D_{max}$  is the maximum number of times that a group serves as the detector group.

### 5.6 Deployment Time Checks

Thus far, we have used deployment locations to determine whether a node should be marked as trusted or not. As noted, it is possible that a substantial number of benign nodes are regarded as untrusted due to errors or inaccuracy in the placement of nodes. This increases the overheads of our schemes. We propose to reduce these overheads by leveraging the time of deployment as a measure of whether a node should be marked as trusted or not. The key idea is to pre-announce the deployment time of each group, and have nodes treat as trusted any node that initiates communications within a short time of its expected deployment.

This involves two important assumptions. First, we assume that it takes some time for an attacker to locate and compromise a sensor node. This need not be a long time, but we assume that there is a minimum amount of time  $T_c$  that it takes to compromise a node once it has been deployed.<sup>1</sup> Second, we assume that the clocks of all nodes are loosely synchronized with a maximum error of  $\epsilon$ . This can be achieved by the use of secure time synchronization protocols as proposed in [29, 40, 66, 67]. We also note that if the network uses occasional broadcasts from the base station, authenticated time stamps could be included with little additional cost. Since time synchronization adds additional overheads, we do not recommend that this method be used unless time synchronization is already needed for some other reason, for example, to ensure correct timestamps on the data collected by sensors.

The deployment time checks can be used to enhance any of the schemes in this section, as they are just another way to decide whether a node can be trusted or not. If any node is trusted due to its location, then it remains trusted despite a late arrival to the network. While this means that replicas could be deployed in the home zone of the original compromised node, this is a limited attack of minimal value to the adversary. However, it would also be possible to require both a correct deployment time and a correct location to be trusted. This would increase security at the cost of additional overheads.

Due to redeployment or supplemental deployment, some nodes may be deployed later than others. If deployment time checks are used, newly joining nodes cannot determine the earlier deployment times of nodes already in the system. Replica nodes could take advantage of this by waiting until new nodes joined the network and asking them to become neighbors. Thus, new nodes should only use location information to determine whether to treat existing nodes as trusted. Note that this

---

<sup>1</sup>According to [34], it took approximately one minute to compromise a node.

implies that deployment time checks cannot be used by themselves and should be used to enhance the performance of the location claim and multi-group approaches.

### 5.6.1 Protocol Description

We now describe the way the deployment time checks are used. When a group  $G_v$  of nodes are deployed, they will be pre-loaded with a time stamp  $T_v$  that is digitally signed by a trusted server. This time stamp indicates that the sensor nodes in  $G_v$  should finish neighbor discovery before time  $T_v$ . If they try to setup neighbor connections with other nodes after time  $T_v$ , they are considered to be untrusted nodes. The time stamp  $T_v$  should be a function of the deployment time  $T$ , the time  $T_c$  needed for compromising and replicating a node, and the maximum time synchronization error  $\epsilon$ . Specifically, the network operator should set  $T + T_n + \epsilon < T_v < T + T_n + T_c - \epsilon$ , where  $T_n$  is the neighbor discovery time, such that no nodes should have clocks too fast to accept the new node, but no new node could be compromised and accepted in time. This means that  $\epsilon < 0.5T_c$  determines the maximum amount of allowable error.

Consider a particular node  $u$ . Assume that its neighbors  $N(u)$  has been marked as trusted or untrusted using the deployment location, according to one of our schemes. We attempt to distinguish more benign nodes from those marked as untrusted by checking the deployment times of those nodes. Specifically, for every node  $v \in N(u)$  that is marked as untrusted, node  $u$  checks whether  $v$  was discovered before  $T_v$ , the deployment time of  $G_v$ . If yes, node  $v$  will be re-marked as trusted.

### 5.6.2 Analysis

The timing-based approach clearly reduces the number of untrusted nodes even if there are large deployment location errors. Let  $P_t$  denote the probability that a

node  $v$  is deployed within time  $T_v$ . The probability that  $v$  is marked as untrusted is  $(1 - P_z) \times (1 - P_t)$ . Thus, we expect  $(1 - P_z) \times (1 - P_t) \times N$  claims instead of  $(1 - P_z) \times N$ , where  $N$  is the total number of sensor nodes. In calculating the communication overhead for Scheme II, we can multiply the probability of a claim from any band  $P_i$  by  $(1 - P_t)$  to derive the likelihood of needing to send a claim from that band. Thus, the communication cost per node is  $C'_s = (1 - P_t)C_s$ . Similarly, the communication cost per node of Scheme III is  $C'_m = (1 - P_t)C_m$ .

The storage and computational overheads per node of Schemes II and III can similarly be estimated as  $2m(1 - P_t)(1 - P_z)$  and  $(1 - P_t)(1 - P_z)(b + 2m)$ ,  $2m(1 - P_t)(1 - P_z)(1 + D_{max})$  and  $(1 - P_t)(1 - P_z)(b + 2m(1 + D_{max}))$ , respectively.

This scheme thus provides substantial reductions in all overheads. For example, if  $P_t = 0.9$ , it reduces all communication, storage, and computational overheads by 90%. In addition, this approach can also enhance network connectivity, since more benign nodes can be correctly re-marked as trusted.

## 5.7 Discussion

In this section, we compare our proposed techniques with existing replica detection methods [58] in terms of the communication, computation, and storage (for location claims) overheads.

Scheme I is the most efficient solution in terms of overheads. In addition, this solution does not require any expensive public key operations or secure localization. We believe that this scheme is suitable to limit replica node attacks in applications that have strict requirements on energy consumption. The potential problem of Scheme I is that the system must trade off between the ability to defend against replica nodes near the group deployment point and the ability of some benign nodes to communicate. The tradeoff depends greatly on accuracy of sensor deployment to the



group deployment points. However, with the help of deployment time checks, we can reduce the sensitivity of the scheme to deployment accuracy such that the loss of communication only occurs when both time synchronization and sensor deployment are inaccurate.

Scheme II provides better security and availability than Scheme I at the cost of additional overheads. Compared with the previously proposed Line-Selected Multicast scheme of [58], Scheme II has numerous advantages. First, Scheme II achieves near-perfect replica detection, while Line-Selected Multicast achieves only about 80% detection in simulation [58]. Both schemes are robust to node compromise, except locally near the replica node, in which case the replica has little practical impact on the network. In terms of overheads, Scheme II is more efficient in communication, computation, and storage. Line-Selected Multicast may use, e.g. six *line segments* per node, each with an expected cost of  $\sqrt{N}$  per line segment [58], where  $N$  is total number of sensor nodes. In the simulation, this was shown to cost approximately  $\sqrt{N}$  packets sent and received per node [58].

As shown in Figure 5.5, Scheme II has at least 86% less communication overhead compared with [58] when  $R_z = 100$  meters. Even when  $R_z = 50$  meters, Scheme II has at least 66% less communication overhead than [58]. Although Scheme III requires more overhead than Scheme II, it still requires less overhead than [58]. In particular, Scheme III saves at least approximately 69% of communication overhead of [58] when  $R_z = 100$  meters. Even when  $R_z = 50$  meters, Scheme III saves approximately 44% as long as  $N \geq 6000$ .

In the Line-Selected Multicast scheme, each node must verify and store  $O(\sqrt{N})$  location claims. Specifically, the Line-selected Multicast scheme requires approximately 100 signatures per node when  $N = 10,000$ , whereas Scheme II can be expected to have less than 20 signatures per node in the sample scenario given in Section 5.4.

Furthermore, the signature verification savings can be significant in the energy-limited lifetime of the sensor nodes. According to the experimental results of [49], iMote2 and TelosB mote sensors consume 3.51 mJ and 21.82 mJ of energy for signature verification, respectively. When using an iMote2, we expect Scheme II to cost less than 70.2 mJ of energy per node for signature verification. Even considering the less power efficient TelosB mote, we would expect a cost per node of less than 436.4 mJ for signature verification. On the contrary, we estimate that Line-Selected Multicast will cost approximately 351 mJ for the iMote2 and 2182 mJ for the TelosB when it requires 100 signature verifications per node.

Adding deployment time checks will significantly reduce the overheads of Scheme II by limiting location claims to replica nodes that are both placed outside their home zones and connect to their neighbors after the allowed time period. If  $P_t = 0.9$ , it reduces the overheads of Schemes II and III by 90%. In general, the more the effort by the network operator into making both the deployment time and deployment locations accurate, the lower the overhead in our schemes. It is not unreasonable to imagine networks in which, for example, the deployment times are sufficiently accurate to enable zero overhead for benign nodes.

## 5.8 Summary

In this chapter, we proposed a set of distributed replica detection schemes for wireless sensor networks that take advantage of the deployment knowledge to reduce overheads while still ensuring high robustness to a powerful adversary. Scheme I is simple and can be effective when the deployment location is accurate and the system does not have strict security requirements. Scheme II achieves high replica detection capability with less communication, computational and storage overheads than prior work. Scheme III provides very strong resilience to node compromise at the cost of

more overheads than Scheme II. Furthermore, the overhead of Schemes II and III can be significantly reduced by using deployment time checks.

The main assumption of our proposed schemes is that nodes can be deployed in groups and that the groups' locations and times of deployment can be estimated in advance. We have argued that these assumptions are reasonable for many systems.

One important advantage of our approach is the possibility of near-zero overheads. By making the deployment times and locations more accurate, the network operator can arbitrarily reduce the overheads of our schemes without loss of security. At the same time, our approach is flexible and robust when there are errors, with costs rising only gradually when more errors are introduced. We showed that the overheads of our schemes are better than existing work for a reasonable range of deployment error estimates, even without assuming deployment time checks. Given these properties, network operators with a broad range of needs can employ our schemes to greatly reduce the threat posed by replica node attacks.

## CHAPTER 6

### MOBILE REPLICAS NODE DETECTION

In Chapter 5, we proposed distributed detection schemes against replica node attacks. Although these schemes efficiently and effectively work in the static sensor networks, these schemes do not work in mobile sensor networks since these schemes rely on fixed sensor locations. To fight against replica node attacks in mobile sensor networks, we propose a fast and efficient mobile replica node detection scheme using the Sequential Probability Ratio Test. To the best of our knowledge, this is the first attempt to tackle the problem of replica node attacks in mobile sensor networks. We show analytically and through simulation experiments that our scheme provides effective and robust replica detection capability with reasonable overheads.

#### 6.1 Overview

In this section, we present an overview of a mobile replica detection scheme.

Several software-based replica node detection schemes have been proposed for static sensor networks [11, 13, 58, 77, 90]. The primary method used by these schemes is to have nodes report *location claims* that identify their positions and attempt to detect conflicting reports that signal one node in multiple locations. However, since this approach requires fixed node locations, it cannot be used when nodes are expected to move. Thus, our challenge is to design an effective, fast, and robust replica detection scheme specifically for mobile sensor networks.

In this chapter, we propose a novel mobile replica detection scheme based on the *Sequential Probability Ratio Test* (SPRT) [72]. We use the fact that an uncom-

promised mobile node should never move at speeds in excess of the system-configured maximum speed. As a result, a benign mobile sensor node's measured speed will appear to be at most the system-configured maximum speed as long as we employ speed measurement system with low error rate. On the contrary, replica nodes will appear to move much faster than benign nodes and thus their measured speeds will likely be over the system-configured maximum speed because they need to be at two (or more) different places at once. Accordingly, if we observe that a mobile node's measured speed is over the system-configured maximum speed, it is then highly likely that at least two nodes with the same identity are present in the network.

By leveraging this intuition, we perform the SPRT on every mobile node using a null hypothesis that the mobile node has not been replicated and an alternate hypothesis that it has been replicated. In using the SPRT, the occurrence of a speed that falls short of or exceeds the system-configured maximum speed will lead to acceptance of the null and alternate hypotheses, respectively. Once the alternate hypothesis is accepted, the replica nodes will be revoked from the network.

We validate the effectiveness, efficiency, and robustness of our scheme through analysis and simulation experiments. Specifically, we find that the main attack against the SPRT-based scheme is when replica nodes fail to provide signed location and time information for speed measurement. To defeat this attack, we employ a quarantine defense technique to block such non-compliant nodes. We then study this technique in two ways. First, we show through quarantine analysis that the amount of time, during a given time slot, that the replicas can impact the network is very limited. Second, we provide a detailed game-theoretic analysis that shows the limits of any attacker strategy over any number of time slots. Specifically, we formulate a two-player game to model the interaction between the attacker and the defender, derive the optimal attack and defense strategies, and show that the attacker's gain is greatly

limited when the attacker and the defender follow their respective optimal strategies. We provide analyses of the number of speed measurements needed to make replica detection decisions, which we show is quite low, and the amount of overhead incurred by running the protocol.

We also evaluate the performance of our scheme using ns-2 simulator. In particular, we consider two types of replicas for performance evaluation: *mobile* and *static*. In case of the mobile replicas, we investigate how replica mobility affects the detection capability of our scheme. In case of the static(immobile) replicas, the attacker keeps his replica nodes close together and immobile to diminish the chance of detection. An exploration of the immobile case is useful in the sense that this case represents the worst case for detection, and thus we can see how our scheme works in the worst case. The simulation results of both cases show that this scheme very quickly detects mobile replicas with low false positive and negative rates.

The rest of chapter is organized as follows. Section 6.2 describes the underlying assumptions for mobile replica detection scheme. Section 6.3 presents the proposed mobile replica detection scheme using the SPRT. Section 6.4 presents the security analysis of the proposed scheme. Section 6.5 presents the performance analysis of the proposed scheme. Section 6.6 presents the results of simulations we conducted to evaluate the scheme. Finally, Section 6.7 summarizes the chapter.

## 6.2 Assumptions

In this section, we describe the network and attacker assumptions of a mobile replica detection scheme.

We consider a two-dimensional *mobile* sensor network where sensor nodes freely roam throughout the network. We assume that every mobile sensor node's movement is physically limited by the system-configured maximum speed,  $V_{max}$ . We also assume

that all direct communication links between sensor nodes are bidirectional. This communication model is common in the current generation of sensor networks.

We assume that an adversary may compromise and fully control a subset of the sensor nodes, enabling him to mount various kinds of attacks. For instance, he can inject false data packets into the network and disrupt local control protocols such as localization, time synchronization, and route discovery process. Furthermore, he can launch denial of service attacks by jamming the signals from benign nodes. However, we place some limits on the ability of the adversary to compromise nodes. If the adversary can compromise a major fraction of the network, he will not need or benefit much from the deployment of replicas. To amplify his effectiveness, he can also launch a replica node attack, which is the subject of our investigation. We assume that the adversary can produce many replica nodes and that they will be accepted as a legitimate part of the network. We also assume that the attacker attempts to employ as many replicas of one or more compromised sensor nodes in the network as will be effective for his attacks. The attacker simply lets replica nodes randomly move or could move his replica nodes in different patterns in an attempt to frustrate our proposed scheme. We discuss this possibility in Section 6.4.

### 6.3 Protocol Description

In this section, we present the details of our technique to detect replica node attacks in mobile sensor networks.

In static sensor networks, a sensor node is regarded as being replicated if it is placed at more than one location. However, if nodes are allowed to freely roam throughout the network, the above technique does not work because the benign mobile node is treated as replica due to the continuous change in locations. Hence, it is imperative to use some other technique to detect replica nodes in mobile sensor

networks. Fortunately, mobility provides us with a clue to help resolve the mobile replica detection problem. Specifically, a benign mobile sensor node should never move faster than the system-configured maximum speed,  $V_{max}$ . As a result, a benign mobile sensor node's measured speed will appear to be at most  $V_{max}$  as long as we employ speed measurement system with low error rate. On the contrary, replica nodes will appear to move much faster than benign nodes and thus their measured speeds will likely be over  $V_{max}$  because they need to be at two (or more) different places at once. Accordingly, if the mobile node's measured speed exceeds  $V_{max}$ , it is then highly likely that at least two nodes with the same identity are present in the network.

Under this intuition, a straightforward approach for replica detection is to decide a mobile node as replica by observing a single evidence that its speed exceeds  $V_{max}$ . However, this approach does not consider the speed measurement error. Due to the error occurrence in the speed measurement, replica (resp. benign) node could be detected as benign (resp. replica) node. To minimize these false positive and negatives, we need to make a decision with multiple pieces of evidence rather than a single evidence. To meet this need, we propose a mobile replica detection scheme based on the Sequential Probability Ratio Test (SPRT) [72]. We believe that the SPRT is well suited for tackling the mobile replica detection problem in the sense that we can construct a random walk with two limits in such a way that each walk is determined by the observed speed of a mobile node; the lower and upper limits are properly configured to be associated with the shortfall and excess of  $V_{max}$ , respectively.

We apply the SPRT to the mobile replica detection problem as follows. Each time a mobile sensor node moves to a new location, each of its neighbors asks for a signed claim containing its location and time information and decides probabilistically whether to forward the received claim to the base station. The base station computes



the speed from every two consecutive claims of a mobile node and performs the SPRT by considering speed as an observed sample. Each time the mobile node's speed exceeds (resp. remains below)  $V_{max}$ , it will expedite the random walk to hit or cross the upper (resp. lower) limit and thus lead to the base station accepting the alternate (resp. null) hypothesis that the mobile node has been (resp. not been) replicated. Once the base station decides that a mobile node has been replicated, it revokes the replica nodes from the network.

Specifically, every sensor node gets the secret keying materials for generating digital signatures before deployment. We will use an identity-based public key scheme [2, 12, 63]. It has been demonstrated that public key operations can be efficiently implemented in static sensor devices [33, 49, 53, 73]. Moreover, most replica detection schemes [13, 58] employ an identity-based public key scheme for static sensor networks. Mobile sensor devices are generally more powerful than static ones in terms of battery power due to the fact that the mobile sensor node consumes a lot of energy to move. Additionally, the energy consumption due to movement is known to be substantially larger than that for public key operations. For instance, the power consumption for the movement of a mobile sensor device was measured as 720 mW [15], whereas the energy consumption for public key signature was measured from 2.9 mW to 48 mW while that for public key verification was measured from 3.5 mW to 58.5 mW in accordance with the sensor hardware platforms [49]. Thus, we believe that the public key scheme can be practical for mobile sensor networks.

We also assume that every mobile sensor node is capable of obtaining its location information and also verifying the locations of its neighboring nodes. This can be implemented by employing mobile node localization methods [39, 70, 83]. Finally, we assume that the clocks of all nodes are loosely synchronized with a maximum error

of  $\epsilon$ . This can be achieved with the help of secure time synchronization protocols [29, 40, 66, 67]. Our proposed protocol proceeds in two phases.

### 6.3.1 Claim Generation and Forwarding

Each time a mobile sensor node  $u$  moves to a new location, it first discovers its location,  $L_u$  and then discovers a set of neighboring nodes,  $N(u)$ . Every neighboring node  $v \in N(u)$  asks for an authenticated *location claim* from node  $u$  by sending its current time  $T$  to node  $u$ . Upon receiving  $T$ , node  $u$  checks whether  $T$  is valid or not. If  $|T' - T| > \delta + \epsilon$ , where  $T'$  is the claim receipt time at  $u$  and  $\delta$  is the estimated transmission delay of claim, then node  $u$  will ignore the request. Otherwise,  $u$  generates location claim  $C_u = \{u || L_u || T || Sig_u\}$  and sends it to a neighboring node  $v$ , where  $Sig_u$  is the signature generated by node  $u$ 's private key. If  $u$  denies the claim requests, or if its claim contains invalid time information or fails to authenticate, then  $u$  will be removed from  $N(v)$ . Also, if  $u$  claims a location  $L_u$  such that the distance between  $L_v$  and  $L_u$  is larger than the assumed signal range of  $v$ , then it will be removed from  $N(v)$ . Once the above filtering process is passed, each neighbor  $v$  of node  $u$  forwards  $u$ 's claim to the base station with probability  $p$ .

### 6.3.2 Detection and Revocation

Upon receiving a location claim from node  $u$ , the base station verifies the authenticity of the claim with the public key of  $u$  and discards the claim if it is not authentic. We denote the authentic claims from node  $u$  by  $C_u^1, C_u^2, \dots$ . The base station extracts location information  $L_u^i$  and time information  $T_i$  from claim  $C_u^i$ . Let  $d_i$  denote the Euclidean distance from location  $L_u^i$  at time  $T_i$  to  $L_u^{i+1}$  at  $T_{i+1}$ . Let

$o_i$  denote the measured speed at time  $T_{i+1}$ , where  $i = 1, 2, \dots$ . In other words,  $o_i$  is defined as:

$$o_i = \frac{d_i}{|T_{i+1} - T_i|} \quad (6.1)$$

Let  $S_i$  denote a Bernoulli random variable that is defined as:

$$S_i = \begin{cases} 0 & \text{if } o_i \leq V_{max} \\ 1 & \text{if } o_i > V_{max} \end{cases}$$

The success probability  $\lambda$  of Bernoulli distribution is defined as:

$$\Pr(S_i = 1) = 1 - \Pr(S_i = 0) = \lambda \quad (6.2)$$

If  $\lambda$  is smaller than or equal to a preset threshold  $\lambda'$ , it is likely that node  $u$  has not been replicated. On the contrary, if  $\lambda > \lambda'$ , it is likely that node  $u$  has been replicated. The problem of deciding whether  $u$  has been replicated or not can be formulated as a hypothesis testing problem with null and alternate hypotheses of  $\lambda \leq \lambda_0$  and  $\lambda \geq \lambda_1$ , respectively, such that  $\lambda_0 < \lambda_1$ . We define a user-configured false positive  $\alpha'$  and false negative  $\beta'$  in such a way that the false positive and negative should not exceed  $\alpha'$  and  $\beta'$ , respectively.

Now we present how the SPRT is performed to make a decision about node  $u$  from the  $n$  observed samples, where a measured speed of  $u$  is treated as a sample. We first define the null hypothesis  $H_0$  and the alternate one  $H_1$  as follows:  $H_0$  is the hypothesis that node  $u$  has not been replicated and  $H_1$  is the hypothesis that  $u$  has been replicated. We then define  $L_n$  as the log-probability ratio on  $n$  samples, given as:

$$L_n = \ln \frac{\Pr(S_1, \dots, S_n | H_1)}{\Pr(S_1, \dots, S_n | H_0)} \quad (6.3)$$

Assume that  $S_i$  is independent and identically distributed. Then,  $L_n$  can be rewritten as:

$$L_n = \ln \frac{\prod_{i=1}^n \Pr(S_i|H_1)}{\prod_{i=1}^n \Pr(S_i|H_0)} = \sum_{i=1}^n \ln \frac{\Pr(S_i|H_1)}{\Pr(S_i|H_0)} \quad (6.4)$$

Let  $\omega_n$  denote the number of times that  $S_i = 1$  in the  $n$  samples. Thus we have

$$L_n = \omega_n \ln \frac{\lambda_1}{\lambda_0} + (n - \omega_n) \ln \frac{1 - \lambda_1}{1 - \lambda_0} \quad (6.5)$$

Where:

$$\lambda_0 = \Pr(S_i = 1|H_0), \quad \lambda_1 = \Pr(S_i = 1|H_1).$$

The rationale behind the configuration of  $\lambda_0$  and  $\lambda_1$  is as follows. On the one hand,  $\lambda_0$  should be configured in accordance with the likelihood of the occurrence that benign node's measured speed exceeds  $V_{max}$  due to the time synchronization and localization errors. On the other hand,  $\lambda_1$  should be configured to consider the likelihood of the occurrence that replica nodes' measured speeds exceed  $V_{max}$ . Since the former likelihood is lower than the latter one,  $\lambda_0$  should be set to lower than  $\lambda_1$ .

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

- $\omega_n \leq \tau_0(n)$  : accept  $H_0$  and terminate the test.
- $\omega_n \geq \tau_1(n)$  : accept  $H_1$  and terminate the test
- $\tau_0(n) < \omega_n < \tau_1(n)$  : continue the test process with another observation.

Where:

$$\tau_0(n) = \frac{\ln \frac{\beta'}{1-\alpha'} + n \ln \frac{1-\lambda_0}{1-\lambda_1}}{\ln \frac{\lambda_1}{\lambda_0} - \ln \frac{1-\lambda_1}{1-\lambda_0}}, \quad \tau_1(n) = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\lambda_0}{1-\lambda_1}}{\ln \frac{\lambda_1}{\lambda_0} - \ln \frac{1-\lambda_1}{1-\lambda_0}}$$

If a mobile node  $u$  is judged as benign, the base station restarts the SPRT with newly arrived claims from  $u$ . If, however,  $u$  is determined to be replicated, the base station terminates the SPRT on  $u$  and revokes all nodes with identity  $u$  from the network.

## 6.4 Security Analysis

In this section, we will first describe the detection accuracy of our proposed scheme and then present attack scenarios to break this scheme and the defense strategy against the attacks. Finally, we will show that attacker's gain is substantially limited by the defense strategy.

### 6.4.1 Detection Accuracy

In the SPRT, two types of errors are defined as follows:

- $\alpha$  : error probability that the SPRT leads to accepting  $H_1$  when  $H_0$  is true.
- $\beta$  : error probability that the SPRT leads to accepting  $H_0$  when  $H_1$  is true.

Since  $H_0$  is the hypothesis that a node  $u$  has not been replicated,  $\alpha$  and  $\beta$  are the false positive and false negative probabilities of the SPRT, respectively. According to Wald's theory [72], the upper bounds of  $\alpha$  and  $\beta$  are obtained as follows:

$$\alpha \leq \frac{\alpha'}{1 - \beta'}, \quad \beta \leq \frac{\beta'}{1 - \alpha'} \quad (6.6)$$

Furthermore, it can be proved [72] that the sum of the false positive and negative probabilities of the SPRT is limited by the sum of user-configured false positive and negative probabilities. Namely, the inequality  $\alpha + \beta \leq \alpha' + \beta'$  holds. Since  $\beta$  is the

false negative probability,  $(1 - \beta)$  is the replica detection probability. Accordingly, the lower bound on the replica detection probability will be:

$$(1 - \beta) \geq \frac{1 - \alpha' - \beta'}{1 - \alpha'} \quad (6.7)$$

From Expressions 6.6 and 6.7, we observe that low user-configured false positive and negative probabilities will lead to a low false negative probability for the sequential test process. Hence, it will result in high detection rates. For instance, if the user configures both  $\alpha'$  and  $\beta'$  to 0.01, then the replica detection is guaranteed with probability 0.99.

#### 6.4.2 Limitations of Replica Node Attacks

Let us now discuss attacks that might be launched by the attacker and the defense strategies against such attacks.

First, a malicious node  $u$  may attempt to forge a claim, either by sending a claim with incorrect data or by sending a claim with a bad signature. However, all of  $u$ 's neighbors will check the validity of  $u$ 's identity, reported location, reported time, and the signature over these values using node  $u$ 's public key. Alternatively,  $u$  can simply ignore the claim requests. In our scheme, if  $u$ 's benign neighbor does not receive a claim despite sending a claim request, it will remove  $u$  from its neighboring set and will not communicate with  $u$ .

We note that if one of  $u$ 's neighbors is malicious, the malicious node can serve as  $u$ 's neighbor for forwarding packets. However, there is little benefit to the attacker of having a replica node in the same area as another compromised node. The compromised node can just as easily report fake data, participate in local control protocols, and eavesdrop on messages sent through it. Furthermore, if the attacker needs one

compromised node to accompany each replica node in the network, there will be a very high cost for replica node attacks. This assumption goes unstated, but is implied by the use of signed location claims in replica detection schemes for static networks as well [13, 58].

Similarly, an attacker will not gain much benefit from having multiple replicas of a single node form a physically close group that always moves together so that they can all claim the same location. This group mobility strategy substantially limits the region affected by the replicas and thus the attacker will not gain much benefit from using the replicas in the limited region. For example, in a false data injection attack, it would be easy to ensure that only one of the replicas' data values at a time is accepted by the data aggregators. Similarly, in local control protocols, only one of the replicas' input values at a time would be taken by their neighbors. In this sense, multiple nodes with the same ID would not have more influence in a region than a single node.

An interesting variant of this attack, however, is to keep replicas close to each other so that the perceived velocity between their location claims is less than  $V_{max}$ . To do this, an attacker coordinates a set of replicas to respond with correct claims only to those claim requests that make it appear as a single node never moving faster than  $V_{max}$ . The attacker can have some replicas group closely together for this purpose; replicas that are further away must ignore claim requests or respond with false claims to avoid detection. To illustrate this attacker's strategy, let us consider a simple attack scenario where a compromised node  $u$  and its replica  $u'$  are fixed to some locations in such a way that the distance between these two nodes is set to  $d$ . We assume that nodes  $u$  and  $u'$  initiate the neighbor discovery process at time  $T_0$  and  $T_0 + \frac{d}{V_{max}}$ , respectively. Moreover, suppose that node  $u$  receives a claim request from neighbor  $v$  at time  $T_0 + \xi$  and node  $u'$  receives request from neighbor  $w$  at time

$T_0 + \frac{d}{V_{max}} + \xi$ . Nodes  $u$  and  $u'$  send claims to  $v$  and  $w$  and ignore all incoming claim requests from other neighbors or give them false claims. Even though either  $v$  or  $w$  may move to a new location, the attacker can control nodes  $u$  and  $u'$  to accept claim requests from newly designated neighbors in such a way that the claim receipt time of  $u$  remains  $\frac{d}{V_{max}}$  time ahead of that of  $u'$ . In this way, the attacker can successfully deceive the base station to believe that  $u$  moves back and forth with speed  $V_{max}$ . This attack scenario can be generalized to the case of a set of replicas and to allow for movement.

Since the replicas do not provide valid claims that would make the observed speed exceed over  $V_{max}$ , they can trick the base station into accepting  $H_0$ , the hypothesis that they are not replicas. To stop this attack, we propose to have the base station check whether each node responds with correct claims to all incoming claim requests.

Specifically, each time a malicious node  $u$  ignores a claim request from a benign neighbor node  $v$  or responds with false claims,  $v$  generates a *denial of claim request notification* message  $DCN = \{v||u||MAC_{K_v}[v||u]\}$  and sends it to the base station, where  $MAC$  stands for message authentication code and  $K_v$  is the shared secret key between  $v$  and the base station. Upon receiving the  $DCN$  message from  $v$ , the base station first checks the authenticity of the  $DCN$  and rejects it if it is invalid. Assume that the entire time domain is divided into time slots. The base station maintains a  $DCN$  counter for each node such that it initializes each counter to 0 and then resets it to 0 at the beginning of each time slot. Each time the base station receives a  $DCN$  message on  $u$  from  $v$ , it increases the  $DCN$  counter for  $u$ . If the  $DCN$  counter for  $u$  exceeds the predefined threshold  $\rho$  during a time slot, it is highly likely that  $u$  discards a substantial fraction of claim requests during a time slot.



In this case, the base station will temporarily *quarantine*  $u$  from the network by disregarding all messages from  $u$  and broadcasting the quarantine information to all nodes. Upon receiving this quarantine message, all nodes will stop communicating with  $u$  except for exchanging claim request and response messages with  $u$ . If the *DCN* counter for  $u$  is sustained at most threshold  $\rho$  during the quarantine period, the base station will release the quarantine that was imposed on  $u$  after the expiry of the quarantine period and broadcast the release information. Otherwise, it will extend the quarantine period by the duration of one time slot. The quarantine period needs to ensure that the replica nodes would be quarantined for longer periods than they would be able to participate freely in the network. This principle will also help prevent frequent oscillation between the quarantine and non-quarantine states. Since the base station determines in each time slot whether to impose quarantine on a node, it can satisfy the above principle by setting the quarantine period to be multiple time slots.

To trick the base station into putting benign nodes into the quarantine, the attacker could send many fake *DCN* messages. Specifically, if the base station receives more than  $\rho$  fake *DCN* messages on  $v$ , the benign node  $v$  will be quarantined even though it conforms to all incoming claim requests. To discourage this type of attack, we restrict each node to send one *DCN* per time slot. If the base station receives more than one *DCN* from a node during a time slot, it will accept only one *DCN* from the node and discard the others. Hence, the attacker needs more than  $\rho$  compromised nodes per time slot to force a benign node to be quarantined, thus suppressing him from mounting fake a *DCN* attack.

#### 6.4.2.1 Short-Term Quantitative Analysis of Quarantine Defense Strategy

Next we quantitatively analyze the limit on the amount of time for which a set of replicas can avoid detection and quarantine when they follow a strategy of responding only to selected claims. Our underlying argument is that the replica nodes must ignore a minimum number of claim requests to avoid detection, but we will configure the quarantine system to react and stop the replica node attacks when many claims are ignored.

Suppose that  $r$  replicas of a compromised node  $u$  are fixed to some locations. We model the arrival of claim requests to each replica as a homogeneous poisson process. Note that the sum of multiple homogeneous poisson processes is also a homogeneous poisson process with the rates summed together. Thus, we model the claim request arrival process of the  $r$  replicas as the homogeneous poisson process with rate parameter  $\theta$ . Note that the interarrival times in the poisson process have an exponential distribution with parameter  $\theta$ . Let  $X_i$ ,  $i = 1, 2, \dots$  be independent, identically distributed (i.i.d.) exponential random variables with parameter  $\theta$ . Here  $X_i$  indicates the  $i$ th interarrival time, which is the time interval between arrivals of  $i$ th and  $(i + 1)$ th claim requests.

Let  $\Delta T$  denote the duration of a time slot. Suppose that claim requests arrive at  $r$  replicas from the beginning to the end of a time slot. Let  $o_i$  be the measured speed between the  $i$ th and the  $(i + 1)$ th claim requests during  $\Delta T$  time. Thus  $o_i$  is defined as  $\frac{d_i}{X_i}$ , where  $d_i$  is the Euclidean distance between a pair of replicas that receive the  $i$ th and the  $(i + 1)$ th claim requests. Let  $p'(d_i)$  be the probability that  $o_i$  exceeds  $V_{max}$  during  $\Delta T$  time, where  $\frac{d_i}{V_{max}} \leq \Delta T$ . Clearly,  $p'(d_i)$  is equivalent to the probability  $\Pr(X_i < \frac{d_i}{V_{max}} \mid X_i \leq \Delta T)$ . Since  $X_i$  follows an exponential distribution

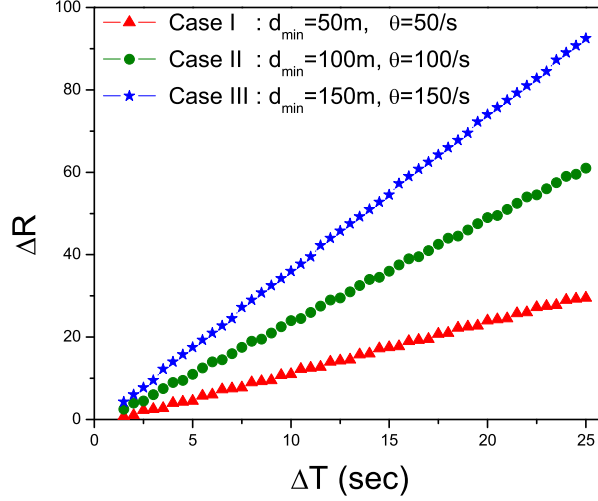
with parameter  $\theta$ , the probability  $p'(d_i)$  is computed with the aid of the improper integral:

$$\begin{aligned}
 p'(d_i) &= \Pr \left( X_i < \frac{d_i}{V_{max}} \middle| X_i \leq \Delta T \right) \\
 &= \frac{\lim_{c \rightarrow \frac{d_i}{V_{max}}} 1 - e^{-c\theta}}{1 - e^{-\theta\Delta T}} \\
 &= \frac{1 - e^{-\frac{\theta d_i}{V_{max}}}}{1 - e^{-\theta\Delta T}}
 \end{aligned}$$

We define  $Y_i$  as a Bernoulli random variable with success probability  $p'(d_i)$  such that:

$$Y_i = \begin{cases} 0 & \text{if } X_i \geq \frac{d_i}{V_{max}} \text{ given } X_i \leq \Delta T \\ 1 & \text{if } X_i < \frac{d_i}{V_{max}} \text{ given } X_i \leq \Delta T \end{cases}$$

Note that  $\mu = \theta \times \Delta T$  is the expected number of events that occur during  $\Delta T$  time in the homogeneous poisson process with rate  $\theta$ . Accordingly, the expected number of claim requests during  $\Delta T$  time is  $\mu$ . By considering the claim forwarding probability  $p$ , the expected number of legitimate claims forwarded to the base station during  $\Delta T$  time is at most  $p\mu$ , corresponding to  $p\mu - 1$  samples. Hence,  $\sum_{i=1}^{p\mu-1} p'(d_i)$  is the expected number of times that  $Y_i = 1$  in  $p\mu - 1$  samples. Let  $d_{min}$  be the shortest distance between a pair of replicas. Since  $p'(d_{min}) \leq p'(d_i)$ , the sum  $\sum_{i=1}^{p\mu-1} p'(d_i)$  should be no less than  $p'(d_{min}) \times (p\mu - 1)$ . Therefore, the expected number of samples that cause the measured speed to exceed  $V_{max}$  during  $\Delta T$  time is bounded from below by  $p'(d_{min}) \times (p\mu - 1)$ . As a consequence, if  $p'(d_{min}) \times (p\mu - 1) > \lceil \tau_1(p\mu - 1) \rceil - 1$ , the replicas should ignore at least  $\Delta R$  claim requests during  $\Delta T$  time in order to prevent them from being detected, where  $\Delta R = p'(d_{min}) \times (p\mu - 1) - \lceil \tau_1(p\mu - 1) \rceil +$

Figure 6.1:  $\Delta R$  vs.  $\Delta T$ .

1. However, if they ignore all of  $\Delta R$  requests, they will be quarantined under the condition of  $\Delta R > \rho$ . In this sense, the replicas are stuck between getting detected and quarantined. Thus, we should set  $\rho$  to be below  $\Delta R$  to ensure that we either detect or quarantine replicas.

Let us now investigate how to configure  $\Delta T$  to hold  $\Delta R > \rho$  under different settings of distance  $d_{min}$  and claim request arrival rate  $\theta$ . For this purpose, we use the following fixed configuration:  $V_{max} = 60$  m/s,  $p = 0.05$ ,  $\lambda_0 = 0.15$ ,  $\lambda_1 = 0.85$ ,  $\alpha' = 0.01$ , and  $\beta' = 0.01$ . We consider three different cases as shown in Figure 6.1. In all three cases,  $\Delta R$  increases linearly with  $\Delta T$ . This implies that  $\Delta T$  needs to be configured in proportion to  $\rho$  to hold  $\Delta R > \rho$ . For instance, if we set  $\rho = 6$ ,  $\Delta T$  should be more than 6, 3, and 2 seconds in Cases I, II, and III, respectively, so as to hold  $\Delta R > \rho$ . This means that the replicas can avoid quarantine during less than  $\Delta T$  time, which is just a few seconds when  $\rho = 6$  in our example scenarios.

Finally, the attacker's only option to avoid detection and quarantine is to move the replicas to an entirely new location before the arrivals of the claim requests that force replicas to be detected or quarantined. However, this will allow the replicas to

escape from detection and quarantine only during less than  $\Delta T$  time, and thus will greatly limit the attacker's ability to control parts of the network for any length of time.

#### 6.4.2.2 Long-Term Game-Theoretic Analysis of Quarantine Defense Strategy

Through the above analysis, we showed that the quarantine defense strategy substantially restricts the attacker's gains from employing selective claim request responding strategy under the circumstance that the duration of a single time slot is reasonably configured. However, the above analysis does not fully reflect the interactions between attacker and defender since it focuses on a single time slot. To analyze how the quarantine strategy provides resilience against the selective claim request responding strategy for a long period of time, we develop a game theoretic model of claim response and quarantine defense. This model is useful to understand what the optimal attack and defense strategies are and how much the attacker's gains are limited by the optimal defense strategy when he employs the optimal attack strategy. In particular, we formulate a game theoretic problem as the two player repeated game with perfect information, where the two players are the attacker and the defender. We believe that the repeated game is suitable for the analysis from the perspective of fully capturing the interactions between the attacker and the defender for a long period of time.

Suppose that attacker deploys  $r$  replicas of a compromised node  $u$  in the network. Let  $N_c$  denote the number of compromised nodes in the network. Let  $\eta$  denote the maximum number of claim requests that can be received by  $u$ 's replicas. Let  $\psi_i$  denote the attacker's replica placement strategy during the  $i$ th time slot. Specifically,  $\psi_i$  indicates the strategy of how the attacker configures the distances between each pair of replicas during the  $i$ th time slot. We define  $\psi$  as attacker's longterm replica

placement strategy consisting of  $\psi_1, \psi_2, \dots, \psi_i, \dots$ . Let  $f(\psi_i)$  denote the fraction of  $\eta$  such that  $0 < f(\psi_i) \leq 1$  when  $\psi_i$  is used. Let  $g(\psi_i)$  denote the ratio of the number of samples that exceed  $V_{max}$  to the total number of samples in the SPRT such that  $0 < g(\psi_i) < 1$  when  $\psi_i$  is used. Recall that  $p$  is the claim forwarding probability. Let  $h(\psi_i)$  denote the number of claim requests that are ignored or ones to which replicas respond with illegitimate claims as to not be detected when  $\psi_i$  is used.  $h(\psi_i)$  is given by

$$h(\psi_i) = \max(g(\psi_i)f(\psi_i)(p\eta - 1) - \lceil \tau_i(f(\psi_i)(p\eta - 1)) \rceil + 1, 0) \quad (6.8)$$

We configure the quarantine threshold  $\rho$  as a positive integer such that  $1 \leq \rho \leq \rho^{max} = \lceil \max h(\psi_i) \rceil - 1$ . The rationale behind  $\rho^{max}$  is to quarantine replicas whose placement achieves the maximum value of  $h(\psi_i)$ . Let  $I(h(\psi_i))$  be a function of  $h(\psi_i)$ . Its output is 0 if  $u$ 's replicas are under quarantine during the  $i$ th time slot. Otherwise, its output is 1 during the  $i$ th time slot. The quarantine period  $q$  is represented as the number of quarantine time slots. If  $h(\psi_i) > \rho$ ,  $u$ 's replicas are quarantined from the  $(i + 1)$ th to the  $(i + q)$ th time slots and thus  $I(h(\psi_j)) = 0$  for  $i + 1 \leq j \leq i + q$ . Each time  $h(\psi_j)$  is more than  $\rho$ , the quarantine period is incremented by one time slot. The pseudo-code for  $I(h(\psi_i))$  is described as Function 1. We model attacker's payoff for a single time slot as the total number of nodes affected by  $u$ 's  $r$  replicas and  $N_c$  compromised nodes. Specifically, we regard a node  $v$  as being affected by  $u$ 's replica if it sends a claim request to  $u$ 's replica and receives a claim from  $u$ 's replica, because  $v$  accepts  $u$ 's replica as its neighbor after receiving a legitimate claim from  $u$ 's replica. In the worst case, in which  $u$ 's replicas receive one claim request per neighbor, the number of nodes that are affected by  $u$ 's replicas is equivalent to the number of claim requests to which  $u$ 's replicas respond with legitimate claims as long as those replicas are neither detected nor quarantined. Since the attacker

---

**Function 1**  $I(h(\psi_i))$ 


---

 INITIALIZATION:  $I(h(\psi_1)) = 1$ 

 VARIABLES:  $k, l$ 

 INPUT:  $h(\psi_i), i \geq 1$ 

OUTPUT: 0 or 1

**if**  $h(\psi_i) \leq \rho$  **then**

   **if**  $I(h(\psi_i)) \neq 0$  **then**

       $I(h(\psi_{i+1})) = 1$ 

   **end if**
**else**

   **if**  $I(h(\psi_i)) \neq 0$  **then**

      **for**  $(k = 1; k \leq q; k++)$ 

        $I(h(\psi_{i+k})) = 0$ 

        $l = i + q + 1$ 

   **else**

       $I(h(\psi_l)) = 0$ 

       $l = l + 1$ 

   **end if**
**end if**


---

will get zero gain for the entire time period if they are detected, he needs to ignore claim requests that contribute to detection or respond with illegitimate claims to those requests while losing as much payoff as the number of ignored or illegitimately responded claim requests. However, if the number of ignored or illegitimately claim requests is more than quarantine threshold  $\rho$  during a time slot, the attacker will obtain zero gain during the quarantined time slots. Thus, he needs to limit the number of ignored or illegitimately responded claim requests in order to reduce his loss incurred by the quarantine defense strategy. By considering these factors, the number of nodes affected by  $u$ 's replicas during the  $i$ th time slot is represented as  $(f(\psi_i)\eta - h(\psi_i))I(h(\psi_i))$  in the worst case. This expression indicates that attacker's gain is the number of claim requests to which  $u$ 's replicas respond with legitimate claims out of the received claim requests during the  $i$ th time slot as long as the number of ignored or illegitimately responded claim requests is at most  $\rho$  in the worst case.

Also, we regard a node  $v$  as being affected by  $\rho$  compromised nodes if it is quarantined due to false DCN messages sent by  $\rho$  compromised nodes. By exploiting the property that quarantined nodes are put under quarantine for at least  $q$  time slots, attacker can make at most  $q$  benign nodes be quarantined per time slot by using  $\rho$  compromised nodes. Accordingly, the total number of affected nodes by  $N_c$  compromised nodes during the  $i$ th time slot is at most  $\frac{qN_c}{\rho}$ .

To model the attacker's payoff for long period of time, we use the limit-of-means payoff as in [71]. This model is useful in the sense that attacker's long term payoff is expressed in terms of the expected payoff per time slot and thus it can converge to a certain value when the number of time slots goes to infinity. We denote the attacker's long term payoff by  $U(\rho, \psi)$  that is defined as:

$$U(\rho, \psi) = \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{i=1}^M (f(\psi_i)\eta - h(\psi_i))I(h(\psi_i)) + \frac{qN_c}{\rho} \quad (6.9)$$

where  $M$  is the number of time slots.

To explore the interactions between the attacker and the defender for long period of time, we formulate a minimax repeated game with  $U(\rho, \psi)$  as follows:

$$\min_{\rho} \max_{\psi} U(\rho, \psi) \quad (6.10)$$

In this game, the strategies of the attacker and the defender are  $\psi$  and  $\rho$ , respectively. The attacker's goal is to maximize his long term payoff  $U(\rho, \psi)$  by controlling the replica placement strategy  $\psi$ . On the other hand, defender's goal is to minimize the maximum value of  $U(\rho, \psi)$  by controlling the quarantine threshold  $\rho$ . The optimal strategies for the attacker and the defender are the ones that make their goals be



achieved. We now solve the above minimax optimization problem to find the optimal strategies for attacker and defender.

Let  $\psi^*$  and  $\psi^{max}$  be the attacker's replica placement strategies, such that

$$\arg \max_{\psi_i} [f(\psi_i)\eta - h(\psi_i)] = \begin{cases} \psi^* & \text{if } 0 \leq h(\psi_i) \leq \rho \\ \psi^{max} & \text{if } h(\psi_i) > \rho \end{cases}$$

Let us define  $w(\psi^*)$  such that  $h(\psi^*) = w(\psi^*)\rho$  and  $0 \leq w(\psi^*) \leq 1$ .

**Lemma 3** *The argument of the maximum of  $U(\rho, \psi)$  as a function of  $\psi$  is  $\psi^*$  under the condition that quarantine period  $q \geq \frac{f(\psi^{max})\eta - h(\psi^{max})}{f(\psi^*)\eta - h(\psi^*)} - 1$ .*

Proof: Let us first denote  $A(\psi_i) = f(\psi_i)\eta - h(\psi_i)$ . Let us consider two cases on a basis of  $\rho$ . In the case that  $0 \leq h(\psi_i) \leq \rho$ ,  $\sum_{i=1}^M A(\psi_i) \leq \sum_{i=1}^M A(\psi^*)$  holds. In the case that  $h(\psi_i) > \rho$ ,  $\sum_{i=1}^M A(\psi_i) \leq \sum_{i=1}^M A(\psi^{max})$  holds. Recall that the quarantine period is  $q$  time slots. If replica placement strategy  $\psi^{max}$  is used in a time slot, replicas are under quarantine for the next  $q$  time slots. Hence, the use of  $\psi^{max}$  virtually results in the consumption of  $q + 1$  time slots. Let us denote by  $\varphi M$  the number of times that  $\psi^{max}$  is used in  $M$  time slots. By considering the general case of including these two cases, the following inequality on  $\sum_{i=1}^M A(\psi_i)$  holds:

$$\sum_{i=1}^M A(\psi_i) I(h(\psi_i)) \leq \varphi M A(\psi^{max}) + (M - \varphi(q + 1)M) A(\psi^*)$$

Therefore, the following inequality on  $U(\rho, \psi)$  also holds:

$$U(\rho, \psi) \leq \varphi(A(\psi^{max}) - A(\psi^*)(1 + q)) + A(\psi^*) + \frac{qN_c}{\rho} \quad (6.11)$$

If  $A(\psi^{max}) < A(\psi^*)(1 + q)$ , the right side of Inequality 6.11 is a strictly decreasing function of  $\varphi$  because its derivative with respect to  $\varphi$  is less than zero and

reaches its maximum when  $\varphi = 0$ . Accordingly,  $U(\rho, \psi) \leq A(\psi^*) + \frac{qN_c}{\rho}$  holds. If  $A(\psi^{max}) = A(\psi^*)(1+q)$ ,  $U(\rho, \psi) \leq A(\psi^*) + \frac{qN_c}{\rho}$  also holds. Hence, if  $q \geq \frac{A(\psi^{max})}{A(\psi^*)} - 1 = \frac{f(\psi^{max})\eta - h(\psi^{max})}{f(\psi^*)\eta - h(\psi^*)} - 1$ ,  $U(\rho, \psi)$  reaches its maximum value when  $\psi_i = \psi^*$  for all  $i \geq 1$ . By Lemma 3,  $\min_{\rho} \max_{\psi_i} U(\rho, \psi)$  is equivalent to  $\min_{\rho} U(\rho, \psi^*)$ . Let us denote  $V(\psi^*)$  such that  $V(\psi^*) = \frac{\eta}{(g(\psi^*) - \frac{\nu_2}{\nu_0})(p\eta - 1)}$ . We also denote  $\nu_0, \nu_1, \nu_2$  such that  $\nu_0 = \ln \frac{\lambda_1}{\lambda_0} - \ln \frac{1-\lambda_1}{1-\lambda_0}$ ,  $\nu_1 = \ln \frac{1-\beta'}{\alpha'}$ ,  $\nu_2 = \ln \frac{1-\lambda_0}{1-\lambda_1}$ .

**Lemma 4** *The argument of the minimum of  $U(\rho, \psi^*)$  as a function of  $\rho$  is  $\rho^*$  under the conditions that  $\frac{\nu_1}{\nu_0} > 1$  and the quarantine period  $q \geq \frac{f(\psi^{max})\eta - h(\psi^{max})}{f(\psi^*)\eta - h(\psi^*)} - 1$ .  $\rho^*$  is given by*

$$\rho^* = \begin{cases} 1 & \text{if } \frac{qN_c}{V(\psi^*)-1} \leq w(\psi^*) \leq 1 \\ 1 & \text{if } \frac{qN_c}{2(V(\psi^*)-1)} \leq w(\psi^*) < \frac{qN_c}{V(\psi^*)-1} \\ \min(m, \rho^{max}) & \text{if } \frac{qN_c}{m(m+1)(V(\psi^*)-1)} \leq w(\psi^*) \text{ and} \\ & w(\psi^*) < \frac{qN_c}{(m-1)m(V(\psi^*)-1)} \\ \rho^{max} & \text{if } w(\psi^*) = 0 \end{cases}$$

for  $m \geq 2$ . Proof: In the case that  $w(\psi^*) = 0$ ,  $U(\rho, \psi^*) = \frac{qN_c}{\rho} + f(\psi^*)\eta$  holds. Since  $\frac{dU(\rho, \psi^*)}{d\rho} < 0$  on the interval  $[1, \rho^{max}]$ ,  $U(\rho, \psi^*)$  is decreasing function of  $\rho$  on the interval  $[1, \rho^{max}]$  and reaches its minimum value at  $\rho = \rho^{max}$  on the interval  $[1, \rho^{max}]$ . Hence,  $\rho^* = \rho^{max}$ .

In the case that  $0 < w(\psi^*) \leq 1$ ,  $U(\rho, \psi^*) = \frac{qN_c}{\rho} + f(\psi^*)\eta - w(\psi^*)\rho$  holds. By using the property that

$$\lceil \tau_1(f(\psi_i)(p\eta - 1)) \rceil = \tau_1(f(\psi_i)(p\eta - 1)) + z(\psi_i)$$

such that  $0 \leq z(\psi_i) < 1$  and the Equation 6.8, we can express  $f(\psi^*)$  as follows:

$$f(\psi^*) = \frac{w(\psi^*)\rho + \frac{\nu_1}{\nu_0} + z(\psi^*) - 1}{(g(\psi^*) - \frac{\nu_2}{\nu_0})(p\eta - 1)} \quad (6.12)$$

By plugging  $f(\psi^*)$  into  $U(\rho, \psi^*)$ , we have

$$U(\rho, \psi^*) = w(\psi^*)(V(\psi^*) - 1)\rho + \frac{qN_c}{\rho} + \frac{\frac{\nu_1}{\nu_0} + z(\psi^*) - 1}{V(\psi^*)}$$

Since the number of claim requests received by replicas is always larger than the number of claim requests to which they respond with legitimate claims as not to be detected when  $\psi^*$  is used,  $f(\psi^*)\eta > h(\psi^*)$  holds. Accordingly,  $V(\psi^*) > 0$  also holds under the condition that  $\frac{\nu_1}{\nu_0} > 1$ . If  $V(\psi^*) = \frac{\eta}{(g(\psi^*) - \frac{\nu_2}{\nu_0})(p\eta - 1)} > 0$ ,  $V(\psi^*) > 1$  holds because  $0 < g(\psi^*) - \frac{\nu_2}{\nu_0} < 1$  and  $0 < p < 1$ . With the property of  $V(\psi^*) > 1$  under the condition that  $\frac{\nu_1}{\nu_0} > 1$ , we consider three sub-cases as follows:

*Sub-case 1:* If  $\frac{qN_c}{V(\psi^*)-1} \leq w(\psi^*) \leq 1$ ,  $\frac{dU(\rho, \psi^*)}{d\rho} \geq w(\psi^*)(V(\psi^*) - 1) - \frac{qN_c}{\rho^2} \geq 0$  holds on the interval  $[1, \rho^{max}]$ . Accordingly,  $U(\rho, \psi^*)$  is increasing function of  $\rho$  on the interval  $[1, \rho^{max}]$  and reaches its minimum value at  $\rho = 1$ . Hence,  $\rho^* = 1$ .

*Sub-case 2:* If  $\frac{qN_c}{2(V(\psi^*)-1)} \leq w(\psi^*) < \frac{qN_c}{V(\psi^*)-1}$ ,  $U(\rho, \psi^*) \geq \frac{qN_c\rho}{2} + \frac{qN_c}{\rho} + \frac{\frac{\nu_1}{\nu_0} + z(\psi^*) - 1}{V(\psi^*)}$  holds. Since  $\frac{dU(\rho, \psi^*)}{d\rho} \geq \frac{qN_c}{2} - \frac{qN_c}{\rho^2} > 0$  on the interval  $[2, \rho^{max}]$  and  $\min U(1, \psi^*) = \min U(2, \psi^*)$ ,  $U(\rho, \psi^*)$  reaches its minimum value at  $\rho = 1$ . Hence,  $\rho^* = 1$ .

*Sub-case 3:* If  $\frac{qN_c}{m(m+1)(V(\psi^*)-1)} \leq w(\psi^*) < \frac{qN_c}{(m-1)m(V(\psi^*)-1)}$ ,  $U(\rho, \psi^*) \geq \frac{qN_c\rho}{m(m+1)} + \frac{qN_c}{\rho} + \frac{\frac{\nu_1}{\nu_0} + z(\psi^*) - 1}{V(\psi^*)}$  holds. Since  $\frac{dU(\rho, \psi^*)}{d\rho} = \frac{qN_c}{m(m+1)} - \frac{qN_c}{\rho^2} > 0$  on the interval  $[m+1, \rho^{max}]$  and  $\min U(m, \psi^*) = \min U(m+1, \psi^*)$ ,  $U(\rho, \psi^*)$  reaches its minimum value at  $\rho = m$ . Since  $\rho$  should be configured to be at most  $\rho^{max}$ ,  $\rho^* = \min(m, \rho^{max})$ .

**Theorem 1** *If we set the quarantine period  $q$  such that  $q \geq \frac{f(\psi^{max})\eta - h(\psi^{max})}{f(\psi^*)\eta - h(\psi^*)} - 1$ , the strategies  $\psi^*$  and  $\rho^*$  are optimal for the attacker and defender, respectively.*

Proof: By Lemmas 3, 4, the arguments of the minimax of  $U(\rho, \psi)$  are  $\psi^*$  and  $\rho^*$  under the condition that  $q \geq \frac{f(\psi^{max})\eta - h(\psi^{max})}{f(\psi^*)\eta - h(\psi^*)} - 1$ . Therefore, the optimal strategies of the attacker and defender are  $\psi^*$  and  $\rho^*$  if  $q$  is at least  $\frac{f(\psi^{max})\eta - h(\psi^{max})}{f(\psi^*)\eta - h(\psi^*)} - 1$ .

Now we examine the characteristics of the functions  $f(\psi_i)$ ,  $g(\psi_i)$ , and  $h(\psi_i)$  in terms of  $\psi_i$ . Assume that the average distance between a pair of replicas in  $\psi_i^a$  is less than one in  $\psi_i^b$ . As the distance between a pair of replicas decreases, the overlapped neighborhood areas between a pair of replicas increases; accordingly the total number of affected nodes decreases. Hence,  $f(\psi_i^a)$  is less than  $f(\psi_i^b)$ . Moreover, the decrease of the distance between a pair of replicas leads to a reduction of the measured speed of replicas, and thus the likelihood that a sample exceeds  $V_{max}$  in the SPRT falls off. Hence, both  $g(\psi_i^a)$  and  $h(\psi_i^a)$  are less than  $g(\psi_i^b)$  and  $h(\psi_i^b)$ .

Next we investigate the limitation on the attacker's gain when defender and attacker adhere to their optimal strategies as follows. In particular, we explore the limitation on the attacker's benefit in accordance with conditions of  $w(\psi^*)$ . In the case that  $w(\psi^*) = 1$ ,  $\rho^*$  is set to 1 and thus  $h(\psi^*) = 1$ . According to the characteristics of  $h(\psi^*)$ , the average distance of a pair of replicas is short in  $\psi^*$  such that  $h(\psi^*) = 1$ , and thus  $f(\psi^*)$  is also a small value. Hence, we have a small value of  $f(\psi^*)\eta - h(\psi^*)$ . The number of nodes affected by compromised nodes will also be limited by  $N_c$ . In the case that  $w(\psi^*) = 0$ ,  $\rho^*$  is set to  $\rho^{max}$  and thus  $h(\psi^*) = 0$ . This leads to small value of  $f(\psi^*)\eta - h(\psi^*)$ . Moreover, the compromised nodes only affect  $\frac{qN_c}{\rho^{max}}$  nodes. In the case that  $0 < w(\psi^*) < 1$ ,  $\rho^*$  is set to a value between 1 and  $\rho^{max}$ . As  $w(\psi^*)$  goes to zero and one,  $\rho^*$  goes to  $\rho^{max}$  and 1 and accordingly  $h(\psi^*) = w(\psi^*)\rho^*$  goes to zero and one, respectively. Therefore,  $h(\psi^*)$  takes on the larger values as  $w(\psi^*)$  is further away from zero and one. Under this intuition, we denote the maximum values of  $w(\psi^*)$  and  $\rho^*$  by  $\phi_1$  and  $\phi_2\rho^{max}$  such that  $0 < \phi_1, \phi_2 < 1$ , respectively. Hence, the maximum value of  $h(\psi^*)$  is denoted by  $\phi_1\phi_2\rho^{max}$ . Since the value of  $h(\psi^*)$  is limited by  $\phi_1\phi_2\rho^{max}$ ,  $f(\psi^*)\eta - h(\psi^*)$  is also limited by replica placement strategy  $\psi^*$  such that  $h(\psi^*) = \phi_1\phi_2\rho^{max}$ . We therefore have only  $\frac{qN_c}{\phi_1\phi_2\rho^{max}}$  nodes affected by  $N_c$  compromised nodes.

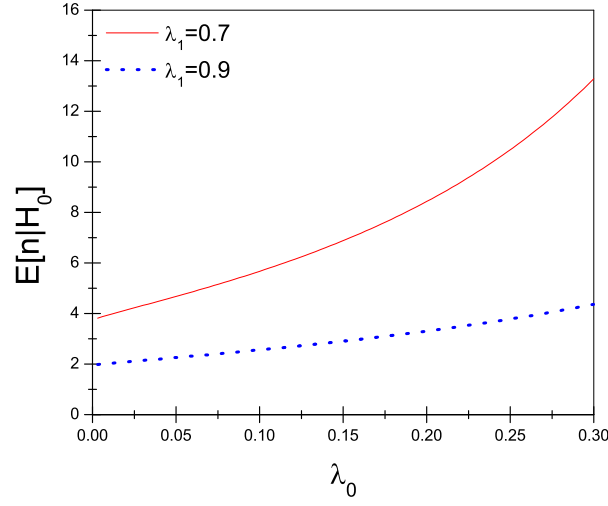


Figure 6.2:  $E[n|H_0]$  vs.  $\lambda_0$  when  $\alpha' = 0.01$  and  $\beta' = 0.01$ .

In all three cases, we see that attacker has limited benefit from employing replicas and compromised nodes when defender and attacker follow their optimal strategies.

## 6.5 Performance Analysis

We now analyze the performance of our scheme in terms of communication, computation, and storage overheads.

### 6.5.1 Communication Overhead

Let us first describe how many observations on an average are required for the base station to make a decision on whether a node has been replicated or not. Then we will present the communication overhead of our scheme.

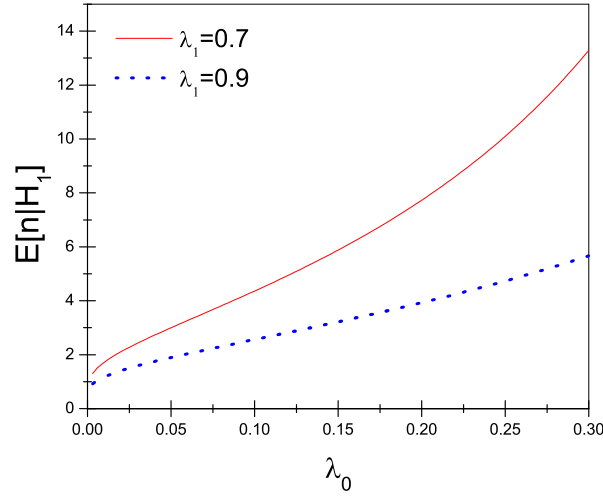


Figure 6.3:  $E[n|H_1]$  vs.  $\lambda_0$  when  $\alpha' = 0.01$  and  $\beta' = 0.01$ .

Let  $n$  denote the number of samples to terminate the SPRT. Since  $n$  varies with the types of samples, it is treated as a random variable with an expected value  $E[n]$ . According to [72],  $E[n]$  is obtained as follows:

$$E[n] = \frac{E[L_n]}{E \left[ \ln \frac{\Pr(S_i|H_1)}{\Pr(S_i|H_0)} \right]} \quad (6.13)$$

From this equation, we compute the expected numbers of  $n$  conditioned on the hypotheses  $H_0$  and  $H_1$  as follows:

$$E[n|H_0] = \frac{(1 - \alpha') \ln \frac{\beta'}{1 - \alpha'} + \alpha' \ln \frac{1 - \beta'}{\alpha'}}{\lambda_0 \ln \frac{\lambda_1}{\lambda_0} + (1 - \lambda_0) \ln \frac{1 - \lambda_1}{1 - \lambda_0}}$$

$$E[n|H_1] = \frac{\beta' \ln \frac{\beta'}{1 - \alpha'} + (1 - \beta') \ln \frac{1 - \beta'}{\alpha'}}{\lambda_1 \ln \frac{\lambda_1}{\lambda_0} + (1 - \lambda_1) \ln \frac{1 - \lambda_1}{1 - \lambda_0}} \quad (6.14)$$

We study how  $E[n|H_0]$  and  $E[n|H_1]$  are affected by the values of  $\lambda_0$  and  $\lambda_1$ . As shown in Figures 6.2 and 6.3,  $E[n|H_0]$  and  $E[n|H_1]$  tend to increase in proportion to  $\lambda_0$  when  $\lambda_1$  is fixed to 0.7 and 0.9, respectively. This implies that the small value of

$\lambda_0$  contributes to decide benign nodes and detect replicas with the small number of claims. When  $\lambda_0$  is fixed,  $E[n|H_0]$  and  $E[n|H_1]$  for the case of  $\lambda_1 = 0.7$  are larger than the corresponding values for the case of  $\lambda_1 = 0.9$ . This means that the large value of  $\lambda_1$  reduces the number of claims required for benign node decision and replica detection.

Now let us compute the communication overhead of our scheme. We define the communication overhead as the average number of claims that are sent or forwarded by nodes in the network. Each time a mobile node  $u$  receives  $b$  claim requests on an average at a location, it sends an average of  $b \times p$  claims to the base station, where  $p$  is the probability that the claim is forwarded to the base station. Let us consider the worst case scenario where every mobile node receives  $b$  claim requests at a location and sends  $b \times p$  claims to the base station at the same time. Since the average hop distance between two randomly chosen nodes is given by  $O(\sqrt{N})$  [58] where  $N$  is the total number of sensor nodes, the communication overhead in the worst case will be  $O(b \times p \times N \times \sqrt{N})$ . Each node's  $b$  claim requests contain the same location information  $L$ . Indeed, the base station needs only one claim per location  $L$ . In this sense,  $b \times p$  can be reduced to one by setting  $p$  to  $\frac{1}{b}$ . Thus, the communication overhead in the worst case scenario can be rewritten as  $O(N\sqrt{N})$  which is equivalent to one of the best scheme proposed in [58].

### 6.5.2 Computation and Storage Overhead

We define computation and claim storage overhead as the average number of public key signing and verification operations per node and the average number of claims that needs to be stored by a node, respectively.

Each time a mobile node receives  $b$  claim requests on an average at a location, it needs to perform  $b$  signature generation operations. Similarly, each time a mobile

node sends  $b$  claim requests on an average at a location, it needs to verify up to  $b$  signatures. In the worst case, every mobile node sends  $b \times p$  claims to the base station at the same time, the base station thus needs to verify up to  $b \times p \times N$  signatures. If  $p$  is set to  $\frac{1}{b}$ , the base station will verify up to  $N$  signatures on an average in the worst case.

The base station stores location claims in order to perform the SPRT whereas every sensor node does not need to keep its or other nodes' claims. Thus, we only need to compute the number of claims that are stored by the base station. In the SPRT, a sample  $o_i$  is obtained from two consecutive location claims of node  $u$ , namely  $C_u^{i-1}$  and  $C_u^i$ . Once a sample  $o_i$  is obtained, the previous location claim  $C_u^{i-1}$  is discarded and current location claim  $C_u^i$  is maintained by the base station. This process is repeated until the SPRT is terminated. Hence, the base station needs to store only one claim per node, so at most  $N$  claims are required to be stored in the base station.

## 6.6 Experimental Study

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

### 6.6.1 Simulation Environment

We simulated the proposed mobile replica detection scheme in a mobile sensor network with the help of the ns-2 network simulator. In our simulation, 500 mobile sensor nodes are placed within a square area of  $500 \text{ m} \times 500 \text{ m}$ .

We use the Random Waypoint Mobility (RWM) model to determine movements of the mobile sensor nodes. In particular, to accurately evaluate the performance of the scheme, we use the RWM model with the steady-state distribution provided by



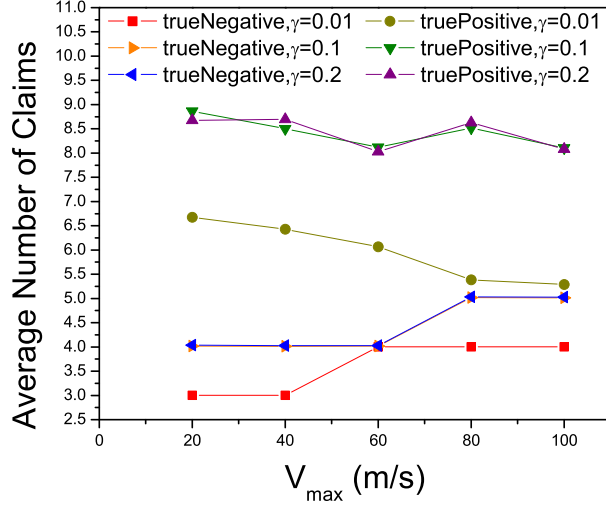


Figure 6.4: Average number of claims vs.  $V_{max}$ .

Random Trip Mobility (RTM) model [3]. We use the codes from [56] to generate RWM model with the steady-state distribution.

All simulations were performed for 1000 simulation seconds. We fixed a pause time of 20 simulation seconds and a minimum moving speed of 1 m/s of each node. Each node uses IEEE 802.11 as the media access control protocol in which the transmission range is 50 m. We set both the user-configured false positive threshold  $\alpha'$  and the false negative threshold  $\beta'$  to 0.01.

To emulate the speed errors caused by the inaccuracy of time synchronization and localization protocols, we modify the measured speeds with speed error rate  $\gamma$ . Specifically, we take speed  $s$  measured using perfect time synchronization and localization protocols and generate speed  $s'$  selected uniformly at random from the range  $[s - s\gamma, s + s\gamma]$ . We evaluated the scheme with  $\gamma$  values of 0.01, 0.1, and 0.2. We set  $\lambda_0$  and  $\lambda_1$  in accordance with  $\gamma$  and  $V_{max}$  as shown in Table 6.1, where L, M, H indicates the low ( $V_{max} = 20, 40$  m/s), moderate ( $V_{max} = 60$  m/s), high ( $V_{max} = 80, 100$  m/s) mobility rates, respectively. The rationale behind the general configurations of  $\lambda_0$  and  $\lambda_1$  has been discussed in Section 6.5. As shown in Table 6.1, these two param-

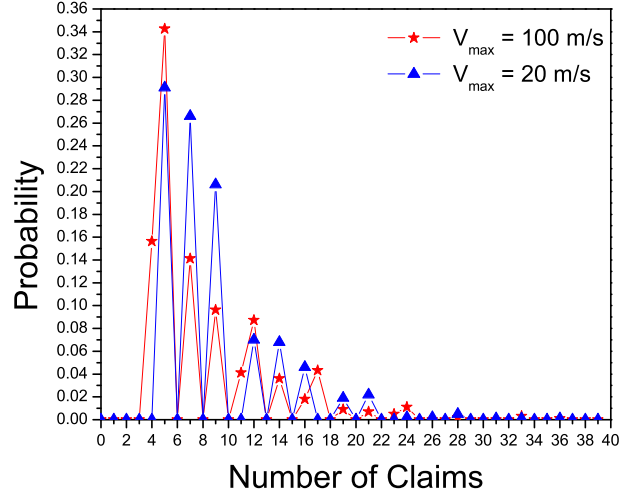


Figure 6.5: Probability distribution of the number of claims when  $\gamma = 0.1$ .

Table 6.1: Parameter Values Used in Simulation Experiments.

$\gamma$	0.01			0.1, 0.2		
Mobility Rate	L	M	H	L	M	H
$\lambda_0$	0.1	0.05	0.01	0.2	0.15	0.1
$\lambda_1$	0.95	0.9	0.8	0.9	0.85	0.8

eters are set in inverse proportion to the rise of mobility rate. The main reason for these configurations is because both mobility and speed error contribute to reduce the chance that a mobile node's speed exceeds  $V_{max}$ .

In our simulation, we consider two cases: *mobileReplica* and *staticReplica*. In the *mobileReplica* case, we use one benign node and one compromised node along with its replica as claim generators. Furthermore, these three nodes' initial placements are randomly chosen and their movements are randomly determined by the RWM model with a steady-state distribution. In the *staticReplica* case, we use one compromised node along with its replica as claim generators. These two nodes do not move while fixing their locations to the initial placements. By studying the *staticReplica* case,

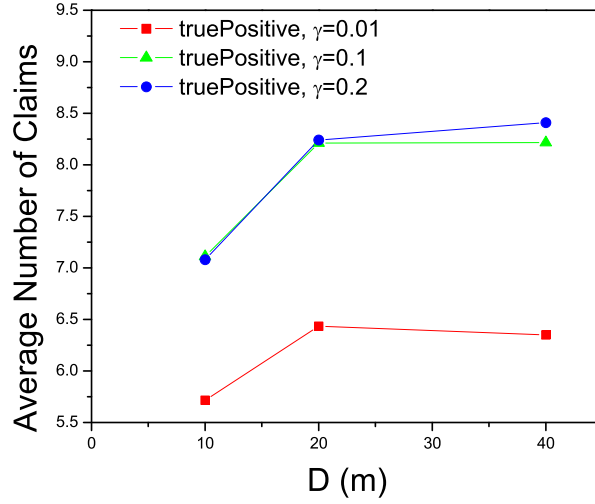


Figure 6.6: Average number of claims vs.  $D$  when  $V_{\max} = 20$  m/s.

we can investigate how the distance between the compromised node and its replica affects the replica detection capability. The staticReplica case represents a strategic attacker and effectively the worst case for detection. The attacker keeps his nodes close together and immobile to lower the chance of detection. As analyzed in Section 6.4, this also limits the attackers's effectiveness. In all scenarios, we assume that all claims that have been forwarded to the base station reach it without any loss. We repeated each simulation scenario 1000 times in such a way that the mobile nodes are initially placed in a different random location each time.

### 6.6.2 Simulation Results

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

- *Number of Claims* is the number of claims required for the base station to decide whether a node has been replicated or not.
- *False Positive* is the error probability that a benign node is misidentified as replica node.

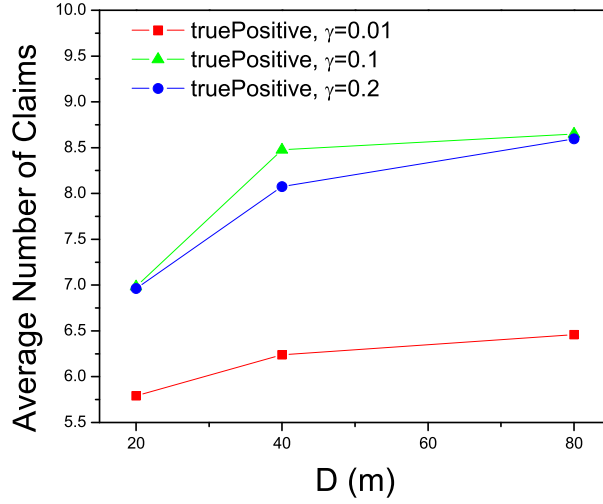


Figure 6.7: Average number of claims vs.  $D$  when  $V_{\max} = 40$  m/s.

- *False Negative* is the error probability that a replica node is misidentified as benign node.

For each execution, we obtain each metric as the average of the results of the SPRTs that are repeated. Note that the SPRT will be terminated if it decides that the claim generator has been replicated. The average of the results of 1000 executions is presented here.

In the experiments, the average number of claim requests,  $b$ , was measured between 17 and 20, depending on  $V_{\max}$ . We associate the configuration of claim forwarding probability  $p$  with  $b$ . Specifically,  $p$  is configured to 0.05 in order to set  $b \times p$  to 1 when  $b$  is assumed to be 20 at all  $V_{\max}$ . The rationale behind this configuration is to make sure that one claim per location is forwarded to the base station on the average.

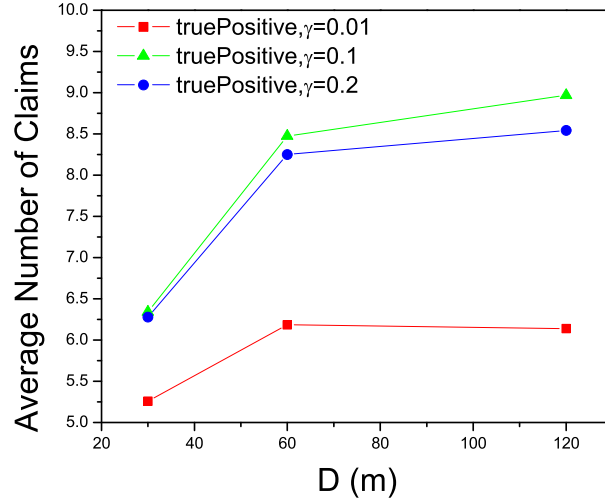


Figure 6.8: Average number of claims vs.  $D$  when  $V_{\max} = 60$  m/s.

#### 6.6.2.1 Mobile Replica Results

In the mobileReplica case, we investigate the false positive and false negative rates and the number of claims while increasing  $V_{\max}$  by 20 m/s in the range from 20 m/s to 100 m/s. The results for the mobileReplica case are summarized as follows.

First, both false positives and false negatives were below 0.01 at all speed error rates and mobility rates. Specifically, the lowest and highest false positives were measured as 0.0, 0.006 when  $\gamma = 0.01, 0.1$  and 0.003, 0.009 when  $\gamma = 0.2$ , respectively. We also observed that there were zero false negatives when  $\gamma = 0.01$ , while the lowest and the highest false negative rates were 0.0, 0.006 when  $\gamma = 0.1, 0.2$ , respectively. Thus, the replica was detected with at least probability of 0.994 and the benign node was misidentified as a replica with at most probability of 0.009 at all speed error rates and mobility rates.

Second, the results of the average number of claims are shown in Figure 6.4. We present the results for two cases. One is that the claim generator is a benign node and the SPRT decides that this node is a benign node. We denote this case by

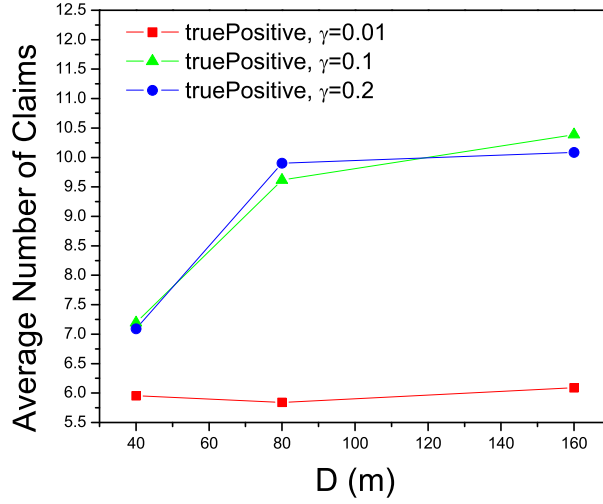


Figure 6.9: Average number of claims vs.  $D$  when  $V_{\max} = 80$  m/s.

*trueNegative* in Figure 6.4. The other case is that the claim generators consist of a compromised node and its replica node, and the SPRT decides that these nodes are a compromised node and its replica. We denote this case by *truePositive* in Figure 6.4.

In the *trueNegative* case, the average number of claims reaches its maximum of 5.03 when  $V_{\max} = 80$  m/s and  $\gamma = 0.2$ . In the *truePositive* case, the average number of claims reaches its maximum of 8.86 when  $V_{\max} = 20$  m/s and  $\gamma = 0.1$ . Thus, the base station reaches correct decisions with a few claims in both cases. Moreover, we see that the average at  $\gamma = 0.1, 0.2$  is higher than that at  $\gamma = 0.01$  in both cases. This indicates that a substantial increase in the speed error rate leads to a rise in the average number of claims.

We also observe that the average number of claims tends to slightly increase and decrease as mobility rate rises in the case of *trueNegative* and *truePositive*, respectively. We infer from this observation that a rise in mobility increases the chance that the speed of a benign node is erroneously measured to be over  $V_{\max}$ , thus delaying the test from moving toward  $H_0$ . On the other hand, a rise in mobility leads to a

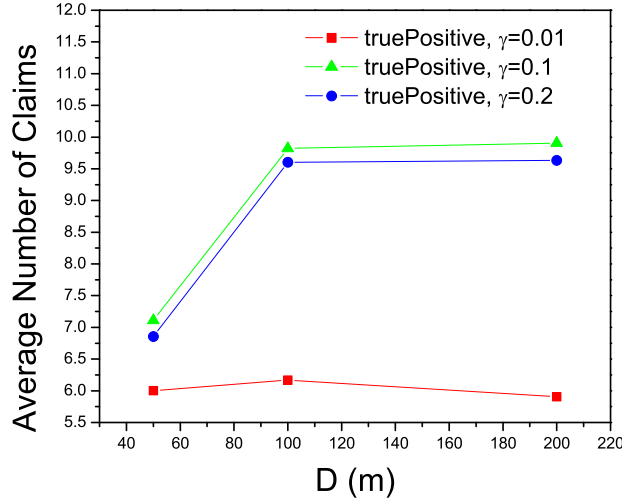


Figure 6.10: Average number of claims vs.  $D$  when  $V_{\max} = 100$  m/s.

reduction in the chance that the replicated node generates the claims containing the same location but different time, and thus expedites moving the test toward  $H_1$ .

Finally, Figure 6.5 shows the probability distribution of the number of claims in the case of truePositive when  $\gamma = 0.1$ . For this distribution, we examine two scenarios: low mobility ( $V_{\max} = 20$  m/s) and high mobility ( $V_{\max} = 100$  m/s). A total of 76.3% and 73.65% of the cases fall in the range from four to nine claims in the case of low and high mobility rates, respectively. This implies that in most cases, the number of claims is less than or close to the average and thus the SPRT detects replicas with at most nine claims in most cases.

#### 6.6.2.2 Static Replica Results

In the staticReplica case, we investigate the same metrics as in mobileReplica case while increasing the distance  $D$  between a compromised node  $u$  and its replica  $u'$  in the range from  $\frac{D_{\max}}{2}$  to  $2D_{\max}$ , where  $V_{\max} = D_{\max}/s$  and the range of  $V_{\max}$  is from 20 m/s to 100 m/s.

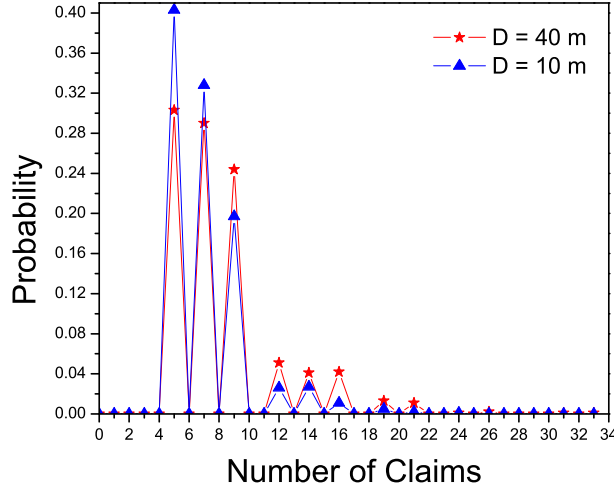


Figure 6.11: Probability distribution of the number of claims when  $\gamma = 0.1$  and  $V_{\max} = 20$  m/s.

First, false negative rates were measured as 0.008 for  $D = 20$  m,  $V_{\max} = 40$  m/s, and  $\lambda = 0.1, 0.2$ . They were 0.0 for all other speed error rates, mobility rates, and  $D$ . Accordingly,  $u$  and  $u'$  were detected with at least 0.992 probability in all cases. Every mobile node checks whether to send claim requests to  $u$  or  $u'$  every 0.5 seconds. Thus, a mobile node's claim request time period is at least 0.5 seconds. Under this claim request time period, we infer from the high detection rate that the inter-arrival time between the claim requests to  $u$  and  $u'$  was highly likely less than  $\frac{D}{V_{\max}}$ . Subsequently, this implies that attacker needs to configure the distance  $D$  in less than  $\frac{D_{\max}}{2}$  under the above claim request time period configuration in order to prevent  $u$  and  $u'$  from being detected with high probability.

Second, Figures 6.6, 6.7, 6.8, 6.9, and 6.10 show the average number of claims when  $V_{\max} = 20, 40, 60, 80, 100$  m/s, respectively. In the truePositive case, the average number of claims is below 10.5 at all mobility rates  $V_{\max}$  and distances  $D$ . Hence, the base station detects  $u$  and  $u'$  with a reasonable number of claims. In terms of the affect of  $\gamma$  on the average number of claims, we see that the substantial increase



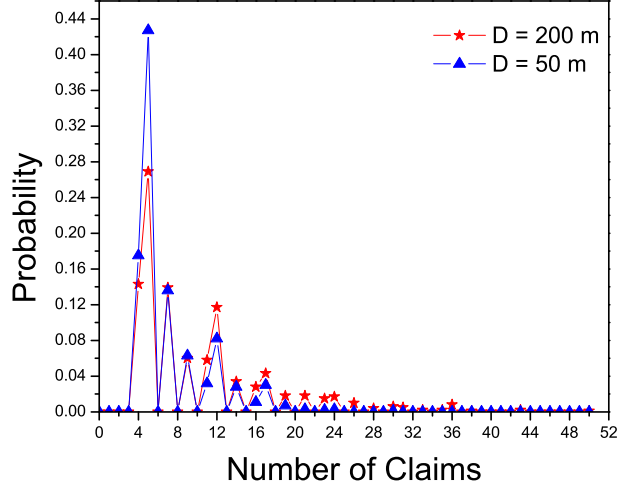


Figure 6.12: Probability distribution of the number of claims when  $\gamma = 0.1$  and  $V_{\max} = 100$  m/s.

of  $\gamma$  from 0.01 to 0.1, 0.2 contributes to a rise in the average number of claims. In terms of the affect of  $D$  on the average number of claims, we observe that the rise of  $D$  from  $\frac{D_{\max}}{2}$  to  $D_{\max}$ ,  $2D_{\max}$  results in an increase of the average number of claims. We infer from this observation that larger values of  $D$  defer the occurrence that the inter-arrival times between the claim requests to  $u$  and  $u'$  exceed  $\frac{D}{V_{\max}}$ , leading to delay in moving the test toward  $H_1$ .

Finally, Figures 6.11 and 6.12 show the probability distribution of the number of claims in truePositive cases when  $\gamma = 0.1$  and  $V_{\max} = 20$  m/s,  $\gamma = 0.1$  and  $V_{\max} = 100$  m/s, respectively. For each case, we examine two scenarios: short distance  $D = \frac{D_{\max}}{2}$  and long distance  $D = 2D_{\max}$ . In case of  $V_{\max} = 20$  m/s, the average number of claims ( $\mu'$ ) and standard deviation ( $\sigma'$ ) in short and long distance scenarios are 7.111 and 2.579, 8.217 and 3.699, respectively. In case of  $V_{\max} = 100$  m/s,  $\mu'$  and  $\sigma'$  in short and long distance scenarios are 7.112 and 3.854, 9.905 and 6.899, respectively. The fraction of the number of claims up to  $\mu' + \sigma'$  is at least 88% and 82% in the case of

$V_{max}=20$  m/s and  $V_{max}=100$  m/s, respectively. This means that in most cases, the number of claims does not exceed  $\mu' + \sigma'$ .

In short and long distance scenarios, a total of 92.8% and 83.7% of the case of  $V_{max}=20$  m/s fall in the range from five to nine claims, respectively. In addition, a total of 80.1% and 61.1% of the case of  $V_{max}=100$  m/s fall in the range from four to nine claims in short and long distance scenarios, respectively. In both cases, we see that the shorter distance  $D$  is, the higher the fraction of claims whose number is at most nine is. We also see that the increase of mobility rate contributes to the decrease of the fraction of the claims whose number is at most nine. Although the number of claims increases in accordance with mobility rate and distance  $D$ , the majority of them are at most 9 in most cases, leading to detection of the static replicas at the cost of a few claims.

## 6.7 Summary

In this chapter, we have proposed a replica detection scheme for mobile sensor networks based on the Sequential Probability Ratio Test (SPRT). Furthermore, we have analytically showed the limitations of the group attack strategy in which the attacker controls the movements of a group of replicas. More specifically, we presented quantitative analysis of the limit on the amount of time for which a group of replicas can avoid detection and quarantine. We also modeled the interaction between the detector and the adversary as a repeated game and solved the game by finding the Nash Equilibrium. We performed evaluations of the scheme under the random movement attack strategy in which the attacker lets replicas randomly move in the network and static placement attack strategy in which he lets replicas fix to their initial placements. The experimental results show that our scheme quickly detects mobile replicas with a small number of location claims in both strategies.

## CHAPTER 7

### WORM PROPAGATION DETECTION

In Chapters 5 and 6, we proposed static and mobile replica node detection schemes to detect and revoke static and mobile replicas in sensor networks. However, attacker needs to have as many sensor nodes as the number of replicas that he wants to employ, resulting in substantial cost due to buying many sensors. To save the cost by replica node attacks, the attacker may attempt to widely spread node compromise by capturing a few nodes and having the captured nodes spread self-propagating worm through the network. As an adversary can quickly compromise an entire sensor network through worm propagation, worms are very dangerous and need to be detected and stopped as quickly as possible. To meet this need, we propose a worm propagation detection scheme in sensor networks. The key idea of the proposed scheme is to adapt the SPRT to tackle the worm propagation detection problem by leveraging the intuition that a worm's communication pattern is different from benign traffic. In particular, a worm in a sensor network requires a long sequence of packets propagating hop-by-hop to each new infected node in turn. We thus have detector nodes observe communication patterns probabilistically sampled from their local regions; a worm spreading hop-by-hop will quickly create chains of connections that would not be seen in normal traffic. To the best of our knowledge, we are the first to propose a scheme for detecting worm propagation in sensor networks. Through analysis and simulation, we demonstrate that the proposed scheme effectively and efficiently detects worm propagation in sensor networks.

## 7.1 Overview

In the sense that finding and physically compromising a large number of nodes would take time and effort and may put the attacker at risk of being discovered, a much better option for the attacker would be to capture and compromise just a few nodes and infect these nodes with self-propagating malware, i.e. with a worm. By simply reintroducing these infected nodes back to the network, the worm could quickly spread and take over a large number of sensors, giving the attacker control over much of the network's operations.

Until recently, however, there was reason to believe that existing wireless sensor networks were immune from such attacks. Sensor motes built on the Harvard architecture keep code and data separate, making software-based exploits like buffer overflows difficult to create. Thus, a worm based on such an exploit should not have been feasible to develop. Francillon et al. [28] showed that this was wrong by demonstrating a remote node compromise attack in which malicious codes are permanently crafted into Harvard-architecture sensor motes and are converted to a self-propagating worm.

This presents a major problem for sensor network security. As we have seen on the Internet, worms can cause tremendous damage [92]. In a sensor network, worm propagation allows the attacker to perform wide-spread node compromise with just a few captured nodes. Hence, worm propagation attacks should be detected and stopped as quickly as possible to minimize the potential for damage in the network.

Although the prevention and detection of worm propagation attacks are essential to stop wide-spread node compromise in sensor networks, little research work has been done regarding this problem. Yang et al. [80] propose a worm propagation prevention scheme based on a software diversity technique. Most sensor motes operate their main software program from flash memory, and we call this program the *flash*

*program*. The main idea of Yang et al. approach is to divide the network into a set of grid cells and assign different version of the flash program to each cell in such a way that two adjacent cells do not share the same version of the flash program. Although a worm can infect one cell by exploiting the vulnerability of the flash program assigned to that cell, it will likely fail to infect the adjacent cells with different versions of flash programs. This approach works under the assumption that each version of flash program has distinct vulnerabilities from all the others. However, this assumption is not always valid due to the following reasons. First, if the attacker finds vulnerabilities in two or more versions of the flash program, a worm will most likely be able to spread to all vulnerable cells. It may be possible to find vulnerabilities in all versions of the program and take over the entire network. Second, although there are ways to diversify software automatically [48], there are no guarantees that the vulnerabilities in one version of the software are different from those in another version. It is unclear that providing such a guarantee is any less difficult than automating the finding and eliminating of security vulnerabilities from software in general. Thus, different versions of the flash program may share a vulnerability such that only variations on the same exploit are required.

To mitigate these limitations, we propose a worm propagation detection scheme for sensor networks using the Sequential Probability Ratio Test (SPRT) in sensor networks. We leverage the following intuition: Since the worm consists of at least 50 packets in the state of the art in sensor worm technology [28], we will very likely observe the patterns of packet propagation when we probabilistically sample a series of worm packets. On the other hand, the amount of benign traffic is relatively limited, both in total and on a per-flow basis; otherwise the costs of normal communication will be very high. Thus, it is unlikely to observe the same patterns of packet propagation when benign traffic is probabilistically sampled at the same rate as in worm. Under

this intuition, the SPRT takes an occurrence of packet repropagation as a sample and dynamically configures the lower threshold and upper threshold in conformity with the type of sample. Under the definition of alternate (resp. null) hypothesis that worm is (resp. not) propagated, the SPRT then starts in the middle of the lower threshold and upper threshold, and accepts the alternate (resp. null) hypothesis if the number of packet repropagations hits or crosses the upper (resp. lower) threshold.

We also propose a biased sampling technique to improve the performance of the SPRT. In biased sampling, the SPRT collects samples more frequently leading to the acceptance of alternate hypothesis than the null one. This results in the reduction of worm propagation detection time and thus greatly limits the number of nodes infected by worm. To the best of our knowledge, our proposed scheme is the first work to detect worm propagation attacks in wireless sensor networks.

The main benefit of our proposed scheme lies in achieving fast worm propagation detection with the aid of the SPRT based on biased sampling. Moreover, it works against any type of worm because it detects worm propagation in line with packet repropagation. Thus, even though the attacker creates and propagates zero-day and polymorphic worms in sensor networks, our proposed scheme still detects the propagations of these worms, while signature-based worm detection schemes would have difficulty in detecting them.

We validate our proposed scheme through analysis and simulation. Specifically, we quantitatively show that our scheme detects worm propagation with high accuracy and greatly limits the attacker's gain from worm propagation. We also show analytically that our scheme requires few samples to make a decision, leading to fast detection of worm propagation. In simulation, we demonstrate that our scheme quickly detects worm propagation with at most six samples while restricting false

positive and false negative rates below at most 0.1% and 1.1%, keeping the fraction of the infected nodes below 12.7% in all cases.

The rest of chapter is organized as follows. Section 7.2 describes the underlying assumptions for our proposed scheme. Section 7.3 describes our proposed scheme using the SPRT. Section 7.4 analyzes the security of our proposed scheme. Section 7.5 analyzes the performance of our proposed scheme. Section 7.6 presents the simulation results for our proposed scheme. Finally, Section 7.7 concludes the chapter.

## 7.2 Assumptions

In this section, we describe the network and attacker assumptions for worm propagation detection scheme.

We assume *static* sensor network in which the locations of sensor nodes are not changed after being placed over the network field. We also assume the bidirectional communication links through which the communicating sensor nodes are able to listen to each other at the same time.

We assume that attacker can physically capture a set of sensor nodes and compromise them. He can also generate worm code and inject it into the compromised nodes, making them worm originators. To model worm propagation in sensor networks, we use the simple epidemic model [16] in which the overall rate of new infections is given by:

$$\frac{dI_t}{dt} = \rho I_t [N - I_t] \quad (7.1)$$

where  $I_t$  is the number of infected hosts at time  $t$ ,  $N$  is the total number of sensor nodes in the network, and  $\rho$  is the pairwise infection rate [16]. In this model, we assume that a sensor has only two states: *susceptible* and *infectious*. All sensor nodes

are initially in the susceptible state except worm originators that are in the infectious state. Once a susceptible node is infected by worm, its state is changed to infectious.

As in [80], we assume that attacker employs a hop-by-hop worm propagation strategy in which infectious node propagates worm to its neighboring susceptible nodes. Attacker can take advantage of this strategy in the sense that he can quickly infect many susceptible nodes through hop-by-hop node contact.

We assume that multiple packets are needed to inject a self-propagating worm into sensor mote. This assumption is very reasonable as the maximum packet size is 28 bytes in current sensor network architectures [28] and thus it is difficult to generate a functional worm fitting to a single packet with such a small size. This assumption is also supported by the state of the art in sensor worm technology [28]. According to the experimental results of [28], a self-propagating worm using 1KByte of memory is implemented and tested in Harvard-architecture sensor motes. When worm is transmitted in its minimum size, the number of packets required to transmit a worm ranges from 50 to 100. This means that at least 50 packets are used to inject a worm into the sensor mote in the most recent version of sensor worm technology.

We also assume that the worm is not propagated through any broadcasting protocol in which each node rebroadcasts messages upon receiving it. This assumption is reasonable in the sense that worm is not malicious code being propagated through the broadcasting protocol but self-propagating malicious code.

### 7.3 Protocol Description

In this section, we describe the details of our scheme for worm propagation detection.

For worm propagation detection in sensor networks, we leverage a simple intuition that the propagation pattern of worm is different to one of benign packets. Our



key assumption is that, in typical sensor network applications, sensors nodes do not frequently send long sequences of packets to each other. We note that doing so would be expensive in terms of power consumption and would shorten the lifetime of the network.

Consider a worm scenario in which a compromised node  $A$  sends a series of worm packets to a benign node  $B$  and infects  $B$  with them. After being infected by  $A$ ,  $B$  will repropagate a series of worm packets to a benign node  $C$  to infect it. Hence, if we probabilistically sample each worm packet from  $A$  to  $B$  and from  $B$  to  $C$ , we likely observe that a worm packet sent from  $A$  to  $B$  is repropagated from  $B$  to  $C$ .

In a benign scenario, suppose that benign node  $A'$  sends a series of benign packets to a benign node  $B$ . We first note that the number of packets is likely to be less than in a worm propagation attack, so probabilistic sampling is less likely to observe this first connection than the worm propagation described above. Also,  $B$  is not going to repropagate these packets to  $C$  after processing them. Note that we do not consider network-level forwarding as propagation for our scheme. More generally,  $B$  does not transmit long sequences very frequently to other sensor nodes, so it is unlikely that probabilistic sampling would observe  $B$  transmitting to  $C$  or any other node right after observing  $A'$  sending to  $B$ . Observing longer chains of benign connections is even more unlikely.

By using this intuition, a simple approach for detection is to observe the number of times that packet repropagation occurs and then compare it to a preset threshold. If it is larger than the threshold, we conclude that a worm is being propagated in the network. This approach will be effective for worm propagation detection as long as the threshold is appropriately configured. However, it is not easy to find a suitable threshold for worm propagation detection. On the one hand, it is likely that the worm is not detected if the system uses a high threshold value, leading to high false negative

rates. On the other hand, it is likely that benign packets will be regarded as worms in the case of a low threshold value, leading to high false positive rates. To reduce false positive and false negatives, we need to dynamically establish the threshold in conformity with the measured number of times that packet repropagation occurs.

To meet this need, we apply the Sequential Probability Ratio Test (SPRT) [72] to the worm propagation detection problem. We benefit from employing the SPRT because it quickly makes a decision with just a few pieces of evidence and with low false positive and false negative rates [72]. In the worm propagation detection problem, we define an alternate (resp. null) hypothesis that the worm is (resp. not) propagated. The SPRT starts in the middle between the alternate hypothesis and null hypothesis and accepts the alternate (resp. null) hypothesis if the number of times that packet repropagation occurs hits or crosses the upper (resp. lower) threshold.

In the pre-deployment stage, the network operator assigns every sensor node a unique ID and secret keying material for pairwise key establishment. A hop-by-hop authentication mechanism is used to prevent source ID forgery. More specifically, when a node transmits a packet to its neighbor, it authenticates the packet with a secret key shared between them. Therefore, upon receiving worm packet, the susceptible node first checks its authenticity and discards it if it is not authentic. This means that a worm packet using a fake source ID will fail to infect susceptible nodes.

After deployment, every sensor node discovers its neighbors and elects itself as a *worm propagation detector* with probability  $p_d$ . Each node repeats this detector selection process periodically. Hence, each node will likely function as a detector in rotation. This random rotation process will prevent the detectors from being exposed to attacker and also help each node save the energy consumption incurred by performing the detector function. We assume that a node is immune to the worm infection when it acts as a detector. This assumption can be easily achieved by having

each detector not act as sink, though it can still act as a data source or relaying node. Although this assumption means that detectors do not receive messages, it will have little effect on the normal operations of the sensor network because we can ensure a small number of detectors by configuring  $p_d$  to a low value and thus a large number of non-detectors can still participate in the data collection process as sinks.

Each time node  $u$  receives a packet from node  $v$ ,  $u$  first performs packet pre-processing as follows. Node  $u$  checks whether the packet is destined for  $u$  if and  $v$  is its neighbor. If so, node  $u$  broadcasts the source and destinations IDs of the received packet to its neighbors with probability  $p_f$ . We call a pair of the source and destination IDs of the received packet *communication pattern*. Upon receiving the communication pattern,  $u$ 's neighbor accepts that pattern if it acts as worm propagation detector. Otherwise, it discards the pattern. The pseudocode for a Packet Preprocessing Unit (PPU) is described as Algorithm 1. In terms of communication layers, PPU resides in the MAC layer. The network program must ensure that the PPU has no vulnerability that can be exploited by an attacker. In other words, the PPU is built as if it were anti-virus/worm software in a computer system. This is reasonable because the PPU consists of a small set of simple statements and thus it is not difficult task to implement the PPU without having any vulnerabilities.

---

**Algorithm 1** Packet Preprocessing Unit (PPU)

---

INPUT: incoming packet  $pkt$

---

**if**  $pkt.destination == u$  and  $pkt.source == u$ 's neighbor **then**  
    broadcast  $\langle pkt.sourceID, pkt.destinationID \rangle$  to neighbors with probability  $p_f$   
**end if**

---

Every detector  $w$  receives the communication patterns probabilistically broadcasted by its neighbors and performs worm propagation detection. More specifically,

$w$  divides the entire time domain into a series of time slots and saves the communication pattern information for one time slot. Each time receives a communication pattern  $(s_i, d_i)$  in given time slot, detector  $w$  inspects whether both  $s_i$  and  $d_i$  are neighbors of  $w$ . If so, it stores  $(s_i, d_i)$  in its memory. Otherwise, it discards  $(s_i, d_i)$ . It then checks whether it already has a communication pattern  $(s_j, d_j)$  such that  $s_i = d_j$ . If so, it creates *merged communication pattern*  $(s_j, s_i, d_i)$  and increases the value of a merged communication pattern counter  $M$ . It also repeats this process when there is a pattern  $(s_l, d_l)$  such that  $d_i = s_l$ , resulting in the creation of merged communication pattern  $(s_i, d_i, d_l)$ . We use the merged communication pattern as the evidence for the occurrence of packet repropagation.

Detector  $w$  maintains  $M$  such that it initializes  $M$  to 0 and then resets  $M$  to 0 at the beginning of each time slot. In a benign scenario,  $w$  is very *unlikely* to generate a merged communication patterns and thus the value of  $M$  is highly likely be zero. In a worm scenario, on the other hand,  $w$  very *likely* generates merged communication patterns and thus the value of  $M$  is highly likely be at least one. Therefore, the values of  $M$  in a series of time slots can be used as evidence of worm propagation in the vicinity of  $w$ . For each time slot, if  $M = 0$ , we set a single sample leading to acceptance of  $H_0$ . Otherwise, we set multiple samples leading to acceptance of  $H_1$  in such a way that it increases the number of samples in line with a rise in the value of  $M$ . We call this sampling strategy *biased sampling*. The main advantage of biased sampling is to generate more samples contributing to acceptance of  $H_1$  rather than  $H_0$  and thus expedite the SPRT to terminate in acceptance of  $H_1$ , resulting in the fast detection of worm propagation and minimizing the number of infected nodes.

Now we describe how to incorporate biased sampling into the SPRT. Let us first denote by  $M_k$  the value of  $M$  in the  $k$ th time slot. We define a Bernoulli random variable  $A_k$ , using  $M_k$  as follows:

$$\begin{aligned} A_k &= 0 \quad \text{if } M_k = 0 \\ A_k, A_{k+1}, \dots, A_{k+M_k-1} &= 1 \quad \text{if } M_k > 0 \end{aligned}$$

We define the success probability  $\gamma$  of the Bernoulli distribution as:

$$\gamma = \Pr(A_k = 1) = 1 - \Pr(A_k = 0) \quad (7.2)$$

Given predefined thresholds  $\gamma_0$  and  $\gamma_1$  such that  $\gamma_0 < \gamma_1$ , if the detector finds that  $\gamma \leq \gamma'$ , it is likely that a worm is not being propagated. On the other hand, if  $\gamma \geq \gamma'$ , it is likely that worm is propagated. The problem of deciding whether a worm is propagated or not can be formulated as a hypothesis testing problem with null and alternate hypotheses of  $\gamma \leq \gamma_0$  and  $\gamma \geq \gamma_1$ , respectively.

Based on this problem formulation, we describe how detector  $w$  runs the SPRT to make a decision about worm propagation from the  $n$  observed samples, where  $A_k$  is treated as a sample. We define  $H_1$  (resp.  $H_0$ ) as the null (resp. alternate) hypothesis that worm is (resp. not) propagated. The log-probability ratio on  $n$  samples  $R_n$  is given as:

$$R_n = \ln \frac{\Pr(A_1, \dots, A_n | H_1)}{\Pr(A_1, \dots, A_n | H_0)}$$

Since merged communication patterns in a time slot occur independently of each other, we can assume that  $A_k$  is independent and identically distributed. Then  $R_n$  can be rewritten as:

$$R_n = \ln \frac{\prod_{k=1}^n \Pr(A_k|H_1)}{\prod_{k=1}^n \Pr(A_k|H_0)} = \sum_{k=1}^n \ln \frac{\Pr(A_k|H_1)}{\Pr(A_k|H_0)} \quad (7.3)$$

Let  $\varphi_n$  denote the number of times that  $A_k = 1$  in the  $n$  samples. Then we have

$$R_n = \varphi_n \ln \frac{\gamma_1}{\gamma_0} + (n - \varphi_n) \ln \frac{1 - \gamma_1}{1 - \gamma_0} \quad (7.4)$$

where  $\gamma_0 = \Pr(A_k = 1|H_0)$ ,  $\gamma_1 = \Pr(A_k = 1|H_1)$ . The rationale behind the configuration of  $\gamma_0$  and  $\gamma_1$  is as follows.  $\gamma_1$  (resp.  $\gamma_0$ ) should be configured in accordance with the likelihood of the occurrence of the merged communication pattern when worm is (resp. is not) propagated.

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

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

We can rewrite the SPRT as follows:

- $\varphi_n \leq t_0(n)$  : accept  $H_0$  and terminate the test.
- $\varphi_n \geq t_1(n)$  : accept  $H_1$  and terminate the test
- $t_0(n) < \varphi_n < t_1(n)$  : continue the test process with another observation.

Where:

$$t_0(n) = \frac{\ln \frac{\beta'}{1-\alpha'} + n \ln \frac{1-\gamma_0}{1-\gamma_1}}{\ln \frac{\gamma_1}{\gamma_0} - \ln \frac{1-\gamma_1}{1-\gamma_0}}, \quad t_1(n) = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\gamma_0}{1-\gamma_1}}{\ln \frac{\gamma_1}{\gamma_0} - \ln \frac{1-\gamma_1}{1-\gamma_0}}$$

If the SPRT terminates in the acceptance of  $H_1$  (resp.  $H_0$ ), detector  $w$  decides that the worm is (resp. is not) propagated within its vicinity. When  $H_0$  is accepted,  $w$  restarts the SPRT. When  $H_1$  is accepted,  $w$  broadcasts to neighboring nodes within a few number of hops a message that there are likely nodes infected by the worm. Then detector  $w$  and these neighbors perform software attestation [55, 79] against their neighbors to detect the nodes infected by worm. The key idea of the software attestation technique is to detect subverted parts in the program codes by checking the integrity of the program codes. If node decides through the attestation that its neighbor has subverted program codes, it will not further communicate with that neighbor.

## 7.4 Security Analysis

In this section, we will first describe the detection accuracy of the proposed scheme and then present the limitations on worm propagation when our proposed scheme is employed. Finally, we will describe the false positives of the proposed scheme.

### 7.4.1 Detection Accuracy

Let us denote  $\alpha$  and  $\beta$  as the false positive rate and false negative rate, respectively. In terms of the SPRT,  $\alpha$  and  $\beta$  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  $\alpha$  and  $\beta$  are given by [72]:

$$\alpha \leq \frac{\alpha'}{1 - \beta'}, \quad \beta \leq \frac{\beta'}{1 - \alpha'} \quad (7.5)$$

where  $\alpha'$  is a user-configured false positive rate and  $\beta'$  is a user-configured false negative rate. According to [72], the summation of  $\alpha$  and  $\beta$  is bounded from above

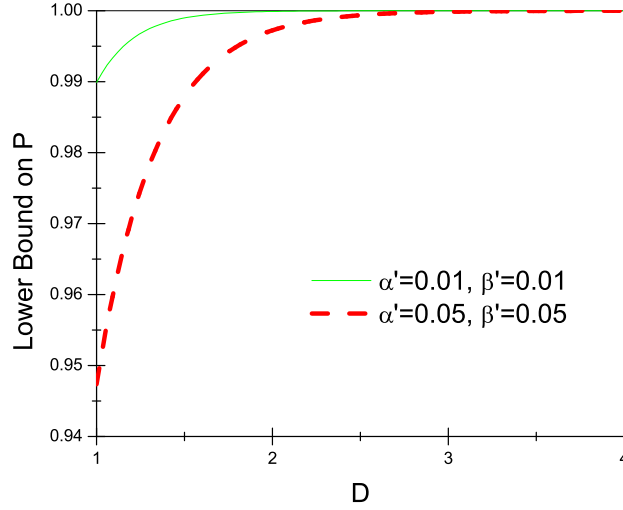


Figure 7.1: Lower bound on worm propagation detection probability  $P$  of  $D$  detectors vs.  $D$  detectors.

by  $\alpha' + \beta'$ . From the above bounds, a lower bound on the worm propagation detection probability of a single detector is given by

$$(1 - \beta) \geq \frac{1 - \alpha' - \beta'}{1 - \alpha'} \quad (7.6)$$

We can infer from Equations 7.5 and 7.6 that low user-configured false positive and false negative rates will result in high worm propagation detection probability where there is a single detector in the network.

We now investigate the worm propagation detection probability of the SPRT, where multiple detectors exist in the network. For this study, we assume that there are  $D$  detectors in the network. From Equation 7.6, we can derive the lower bound on the worm propagation detection probability  $P$  of  $D$  detectors as follows:

$$P = 1 - \beta^D \geq 1 - \left( \frac{\beta'}{1 - \alpha'} \right)^D \quad (7.7)$$



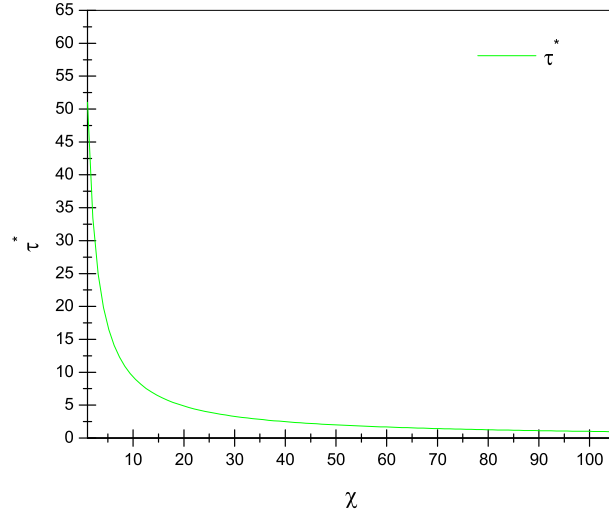


Figure 7.2: The effect of  $\chi$  on  $\tau^*$ .

We investigate how  $D$  affects the lower bound on  $P$  in two cases of  $\alpha' = \beta' = 0.01$  and  $\alpha' = \beta' = 0.05$ . As shown in Figure 7.1, the lower bound on  $P$  increases in accordance with  $D$  and reaches almost 1.0 when  $D = 2$  in the first case and  $D = 3$  in the second case. This means that the SPRT achieves high worm propagation detection capability even if there are only a few detectors in the network.

#### 7.4.2 Limitations on Worm Propagation

We investigate how much our proposed scheme restrains the damage incurred by the worm propagation.

**Lemma 5** *Let us consider the  $n$  time slots of detector  $w$ . Assume that worm is propagated in the vicinity of  $w$  for  $\tau$  time slots out of  $n$  time slots. Let us express by  $\tau\chi$  the number of samples that  $A_k = 1$  in the  $\tau$  time slots for which worm propagation occurs. Worm propagation is detected by  $w$  under the conditions that  $\frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\gamma_0}{1-\gamma_1}}{\chi \ln \frac{\gamma_1}{\gamma_0} - \ln \frac{1-\gamma_1}{1-\gamma_0}} = \tau^* \leq \tau \leq n$  and  $\chi \geq 1$ .*

Proof: Since there are  $\tau\chi$  samples for which  $A_k = 1$ , detector  $w$  sets  $\varphi_n = \tau\chi$ . If  $\varphi_n = \tau\chi \geq t_1(n)$  holds, then  $w$  terminates the SPRT in acceptance of  $H_1$ . Since  $\tau \leq n$  and  $t_1(n) = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\gamma_0}{1-\gamma_1}}{\ln \frac{\gamma_1}{\gamma_0} - \ln \frac{1-\gamma_1}{1-\gamma_0}}$ ,  $w$  detects worm propagation in its vicinity if  $\frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\gamma_0}{1-\gamma_1}}{\chi \ln \frac{\gamma_1}{\gamma_0} - \ln \frac{1-\gamma_1}{1-\gamma_0}} \leq \tau \leq n$  holds.

Let us study how  $\chi$  affects  $\tau^*$  when  $\alpha' = \beta' = 0.01$ ,  $\gamma_0 = 0.1$ ,  $\gamma_1 = 0.9$ , and  $n = 100$ . As shown in Figure 7.2, we observe that  $\tau^*$  sharply decreases as  $\chi$  increases and reaches one when  $\chi = 101$ . This indicates that our scheme greatly limits the number of time slots during which the worm can avoid being detected as the number of samples leading to acceptance of  $H_1$  increases. In particular, our scheme allows a worm to propagate for at most one time slot out of  $n = 100$  time slots when  $\chi$  is 101. Hence, attacker will take little benefit from deploying a worm under the presence of our scheme.

#### 7.4.3 False Positives

We now briefly look into the effect of the false positives on the system. In the SPRT, the false positive rate is bounded above by  $\frac{\alpha'}{1-\beta'}$  and thus maintained to a low value as long as  $\alpha'$  and  $\beta'$  are set to a low value, such as 0.01. Furthermore, the likelihood of observing a merged communication pattern due to benign traffic will be substantially lower than observing one due to worm traffic, leading to a low false positive rate.

In our proposed scheme, communication pattern is transmitted in plaintext. Thus, compromised nodes can generate and send fake communication patterns that contribute to occurrence of the merged communication patterns. This results in false worm propagation detection and occurs false positive case. However, once worm propagation is detected, the compromised nodes will be attested by their neighbors,

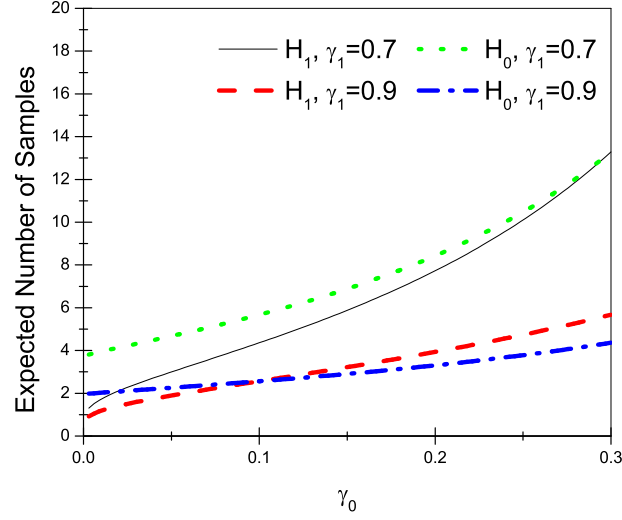


Figure 7.3: The affects of  $\gamma_0$  and  $\gamma_1$  on the expected number of samples conditioned on  $H_0$  and  $H_1$ .

detected and isolated from the network. Therefore, this type of attack will give adverse effects to attacker in terms of facilitating the detection of the compromised nodes.

## 7.5 Performance Analysis

In this section, we first compute how many samples are required on average for a detector to decide whether worm is being propagated or not. We then present the communication and storage overheads of our proposed scheme.

### 7.5.1 Average Number of Samples for Decision

We denote  $n$  by the number of samples that are required for the SPRT to terminate.  $n$  is considered to be a random variable because it is determined in accordance with the types of samples. According to [72],  $E[n]$  is computed as follows:

$$E[n] = \frac{E[R_n]}{E \left[ \ln \frac{\Pr(A_k|H_1)}{\Pr(A_k|H_0)} \right]} \quad (7.8)$$

From Equation 7.8, we calculate  $E[n]$  conditioned on hypotheses  $H_0$  and  $H_1$  as follows:

$$\begin{aligned} E[n|H_0] &= \frac{(1 - \alpha') \ln \frac{\beta'}{1 - \alpha'} + \alpha' \ln \frac{1 - \beta'}{\alpha'}}{\gamma_0 \ln \frac{\gamma_1}{\gamma_0} + (1 - \gamma_0) \ln \frac{1 - \gamma_1}{1 - \gamma_0}} \\ E[n|H_1] &= \frac{\beta' \ln \frac{\beta'}{1 - \alpha'} + (1 - \beta') \ln \frac{1 - \beta'}{\alpha'}}{\gamma_1 \ln \frac{\gamma_1}{\gamma_0} + (1 - \gamma_1) \ln \frac{1 - \gamma_1}{1 - \gamma_0}} \end{aligned} \quad (7.9)$$

Figures 7.3 shows how  $\gamma_0$  and  $\gamma_1$  affect  $E[n|H_0]$  and  $E[n|H_1]$  when  $\alpha' = \beta' = 0.01$ . We see that the SPRT reaches a decision with a small number of samples when  $\gamma_0$  and  $\gamma_1$  are set to a small value and a large value, respectively. We also observe that  $E[n|H_0]$  is larger than  $E[n|H_1]$  when  $\gamma_0$  is a small value. However, as the  $\gamma_0$  increases, the gap between  $E[n|H_0]$  and  $E[n|H_1]$  decreases and eventually  $E[n|H_1]$  exceeds  $E[n|H_0]$ . This means that a small value of  $\gamma_0$  contributes to the faster termination of the SPRT in acceptance of the alternate hypothesis than the null one.

### 7.5.2 Communication and Storage Overheads

We define communication overhead as the average number of communication patterns that need to be sent and received in the network per time slot. Assume that there are  $N$  sensor nodes in the network. There are accordingly  $N \times p_d$  detectors in the network, where  $p_d$  is the detector selection probability. Furthermore, assume

that every sensor node  $u$  has  $b$  neighbors on average and receives on average  $m$  packets that are destined for it per time slot. Node  $u$  on average sends  $m \times p_f \times p_n$  communication patterns to the detectors, where  $p_n$  is the probability that the source of packet is  $u$ 's neighbor. The communication overhead is then on average computed as  $N(p_d \times b + 1) \times m \times p_f \times p_n \approx O(mN)$ . This is reasonable overhead in the sense that every sensor node needs to send and receive on average  $O(1)$  communication pattern per packet received by it.

We define storage overhead as the average number of communication patterns that need to be stored by a sensor node in a time slot. In the worst case, each detector receives and stores at most  $\binom{b}{2}$  communication patterns per time slot. Note that every node acts as detector in rotation. Therefore, the number of communication patterns that need to be stored by each node in a time slot is at most  $f_d \times \binom{b}{2}$ , where  $f_d$  is a fraction of detector duty time slots in the lifetime of each node. Consequently, the storage overhead of the worst case is computed as  $\frac{f_d \times b(b-1)}{2}$ . If  $f_d$  is configured to a low value such as 0.1,  $\frac{b \times f_d}{2}$  can be a small constant when compared to  $b$ . Hence, the storage overhead of the worst case can be approximately rewritten as  $O(b)$ .

## 7.6 Experimental Study

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

### 7.6.1 Simulation Environment

We developed a simple simulation program to evaluate our proposed scheme. In our simulation, we place 1000 sensor nodes within a square area field of 1000 m  $\times$  1000 m and set the communication radius of a sensor node to 50 m. We adopt

a group deployment strategy in which a group of sensor nodes is placed in the field according to the following two-dimensional Gaussian distribution:

$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_g)^2 + (y-y_g)^2}{2\sigma^2}} \quad (7.10)$$

where  $(x_g, y_g)$  is group deployment point and  $\sigma$  is the standard deviation. In group deployment, we set the number of groups and the number of nodes in a group to 20 and 50, respectively. We also set  $\sigma$  to 50, which is equivalent to the communication radius of a sensor node.

We divide the entire time domain into 100 time slots and perform the simulation in units of time slots. We run the SPRT every three time slots. In other words, a time slot in the SPRT corresponds to three time slots in the simulation. The main rationale behind this configuration is to let the SPRT to collect enough pieces of evidence that are useful for worm propagation detection.

We evaluate our scheme in two cases: *benign* and *worm*. In the benign case, nodes only send benign packets. In the worm case, nodes only send worm packets. The main rationale behind this case separation is to investigate the accurate worm propagation detection capability and false positive rates of our scheme without being affected by benign packets and the worm packets, respectively. In the benign case, we assume that benign traffic is generated according to a Poisson distribution and accordingly the inter-arrival times of benign packets follow the exponential distribution with rate parameter  $\lambda$ . Every sensor node sends benign packets to  $\delta$  randomly chosen destinations in such a way that it first selects at uniformly random the start time slot for each destination and then begins to send packets to each destination from the chosen start time slot. To compute the number of packets that are sent per time slot, we use the inverse cumulative distribution function  $Q(t) = \frac{-\ln(1-f_t)}{\lambda}$ , where  $f_t$  is

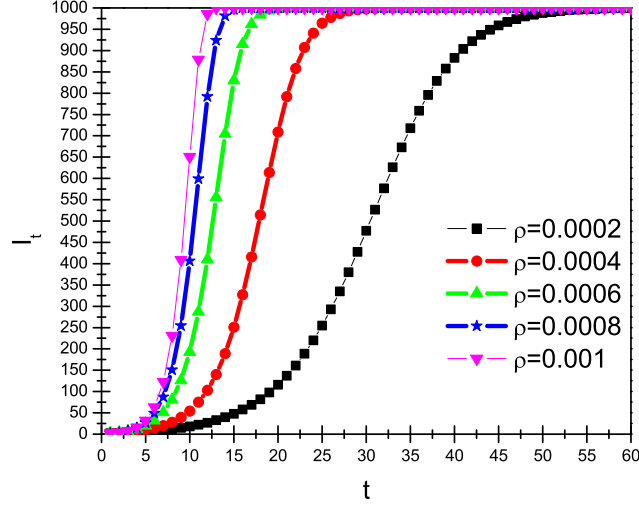


Figure 7.4: Cumulative number of infected nodes  $I_t$  vs. time slot  $t$ .

a fraction of the number of packets from the start time slot to the  $t$ th time slot out of the maximum number of packets such that  $0 \leq f_t < 1$  and it is chosen at uniformly random in each time slot. Thus,  $Q(t) - Q(t-1)$  is the number of packets that are sent in the  $t$ th time slot. We consider five combinations for the configurations of  $\lambda$  and  $\delta$  as follows:  $(\lambda, \delta) = (0.5, 5), (1.0, 10), (1.5, 15), (2.0, 20), (2.5, 25)$  The main reason for these configurations is to maintain reasonable traffic overhead by increasing the number of destinations and decreasing the number of packets to be generated.

In the worm case, we assume that a worm consists of 50 packets. This is reasonable because the minimum worm size is 50 packets in the state-of-art sensor worm technology [28]. As mentioned in Section 7.2, we assume a hop-by-hop worm propagation strategy in which a node repropagates the worm to its neighboring nodes after being infected. We use the discrete-time version [16] of the simple epidemic model to determine the number of infected nodes per time slot. In this discrete-time

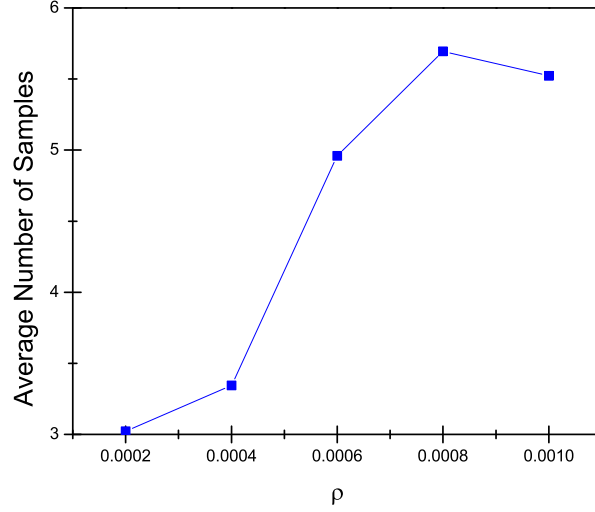


Figure 7.5: Average number of samples vs. pairwise infection rate  $\rho$  in truePositive case.

model, the number of infected nodes is determined in unit of time slot according to the following equation.

$$I_t = (1 + \rho N)I_{t-1} - \rho I_{t-1}^2 \quad (7.11)$$

where  $t$  is the time slot,  $\rho$  is the pairwise infection rate, and  $N$  is the total number of nodes in the network.  $I_0$  indicates the number of compromised nodes before worm propagation begins. In other words,  $I_0$  represents the number of worm originators. We consider a single worm originator and thus  $I_0$  is set to one.  $I_t$  is the cumulative number of infected nodes from the 0th time slot to the  $t$ th time slot. Therefore, the number of infected nodes in the  $t$ th time slot is  $I_t - I_{t-1}$ . We set  $\rho$  from 0.0002 to 0.001 by incrementing 0.0002. Figure 7.4 shows how  $I_t$  grows over time when the worm is propagated without detection in the discrete-time model. We see that  $I_t$  increases in line with the rise of  $t$ . We also observe that the rise of  $\rho$  results in the reduction of the number of time slots required for all sensor nodes to be infected.



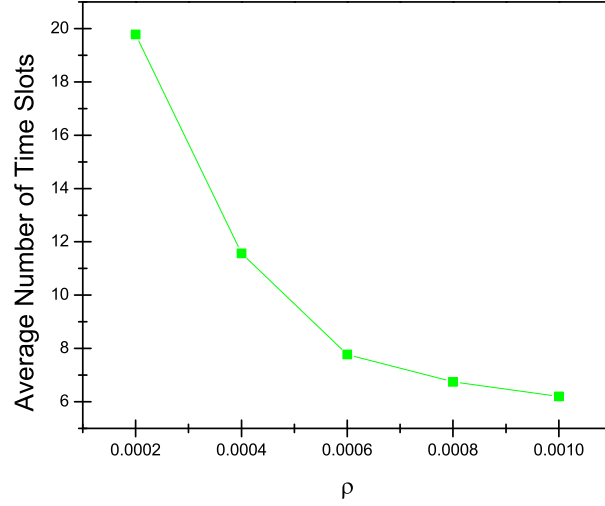


Figure 7.6: Average number of time slots vs. pairwise infection rate  $\rho$  in truePositive case.

We configure detector selection probability  $p_d = 0.1$  and communication pattern forwarding probability  $p_f = 0.02$ . As the worm is at least 50 packets,  $p_f$  is configured to ensure that one packet per worm is forwarded to detectors on average. We also set both the user-configured false positive threshold  $\alpha$  and the false negative threshold  $\beta$  to 0.01, and we set the lower threshold  $\gamma_0$  and the upper threshold  $\gamma_1$  to 0.1 and 0.9, respectively. The rationale behind these configurations is discussed in Section 7.5.

### 7.6.2 Simulation Results

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

- *Number of Samples* is the number of samples required for a set of detectors to make a decision of the existence of worm propagation.
- *Number of Time Slots* is the number of time slots required for a set of detectors to detect worm propagation.
- *Number of Infected Nodes* is the number of infected nodes when worm propagation is detected.

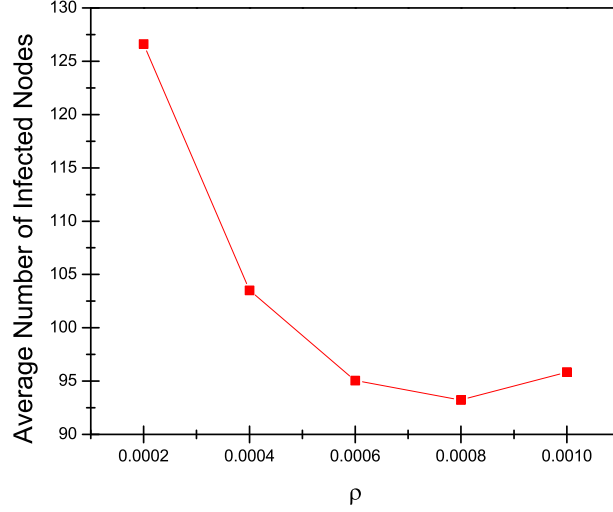


Figure 7.7: Average number of infected nodes vs. pairwise infection rate  $\rho$  in truePositive case.

- *False Positive* is the error probability that a set of detectors misidentifies benign packet transmission as worm propagation.
- *False Negative* is the error probability that a set of detectors misidentifies worm propagation as benign packet transmission.

We present the average results for 1000 runs of the simulation in each configuration such that each run is executed for 100 time slots. For each run, we acquire each metric as the average of the results of the SPRTs that are performed. Note that the SPRT will restart (resp. terminate) if it accepts the null (resp. alternate) hypothesis  $H_0$  (resp.  $H_1$ ). Now we present our main findings.

First, we find that there are no false positives except when  $\lambda = 1.0$  and  $\delta = 10$ , in which there are 0.1% false positives. We also measure the highest false negatives rate as 1.1% when  $\rho = 0.0002$ . Thus, worm propagation is detected with at least 0.989 probability and benign packet transmission is misidentified as worm propagation with at most 0.001 probability. This indicates that the SPRT reaches a decision with very high accuracy.

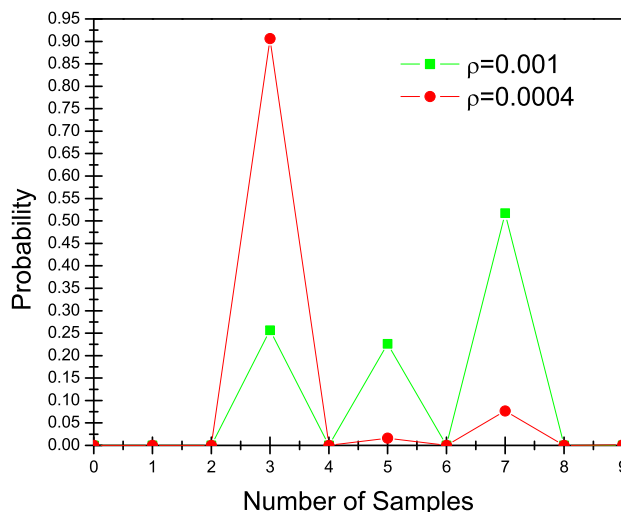


Figure 7.8: Probability distribution of the number of samples.

Second, we define two cases to present results of the average number of samples. One case is that a set of detectors decides that benign packet transmission is benign. We call this case *trueNegative*. The other case is that a set of detectors decides that worm propagation as worm propagation. We call this case *truePositive*. The results of the average number of time slots and infected nodes are only applicable to the *truePositive* case. In the *trueNegative* case, the average number of samples is 3.0 for all settings of  $(\lambda, \delta)$ . This indicates that the SPRT requires only three samples to reach a correct decision on benign packet transmission, irrespective of the values of  $(\lambda, \delta)$ .

In the *truePositive* case, as shown in Figure 7.5, the average number of samples reaches its maximum of 5.695 when  $\rho = 0.0008$ . This means that the SPRT fulfills fast worm propagation detection with at most six samples on average. As shown in Figure 7.6, average number of time slots reaches its maximum of 19.780 when  $\rho = 0.0002$ , given 100 time slots. As shown in Figure 7.7, average number of infected nodes reaches its maximum of 126.617 when  $\rho = 0.0002$ , given 1000 nodes. This

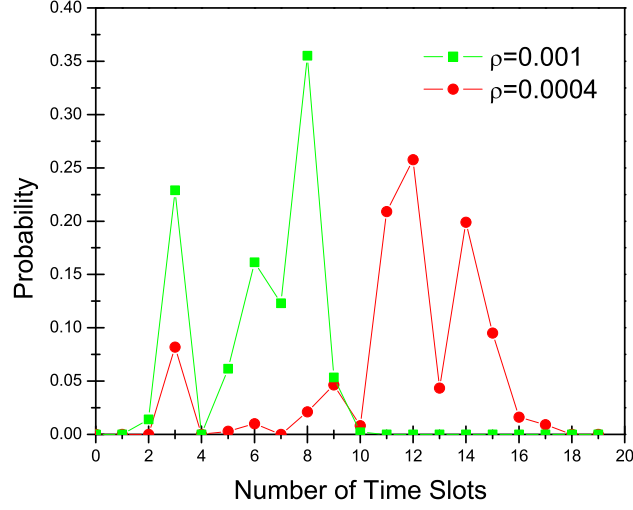


Figure 7.9: Probability distribution of the number of time slots.

indicates that the SPRT substantially restrains the number of time slots and infected nodes below at most 20 and 127, respectively, leading to minimize the damage incurred by worm propagation. We also observe that the average number of time slots and infected nodes tends to decrease as  $\rho$  increases. This is mainly because a rise in  $\rho$  leads to an increase in the number of merged communication patterns, and thus it expedites the SPRT to terminate in the acceptance of  $H_1$ . This results in a reduction in the number of time slots and infected nodes.

Finally, Figures 7.8 and 7.9 show the probability distributions of the number of samples and time slots in case of truePositive. For these distributions, we look into two cases of a low infection rate  $\rho = 0.0004$  and a high infection rate  $\rho = 0.001$ . In cases of low and high infection rate, we see that in 90.6% and 100% of cases, the number of samples is at most 3 and 7, respectively. This means that the number of samples in most cases is below or very close to the average and thus the SPRT performs worm propagation detection with a few number of samples in most cases. We also observe that in 88% and 94.5% of cases, the number of time slots is in

at most 14 and 8, respectively. This indicates that the number of time slots is also very close to the average in most cases and thus the SPRT detects worm propagation quickly.

## 7.7 Summary

In this chapter, we proposed a SPRT-based scheme to detect worm propagation in sensor networks. Furthermore, we analyzed our scheme and show that it detects worm propagation with very high accuracy and few number of samples, and substantially restricts the attacker's gain from worm propagation attacks. Finally, we evaluated the scheme through simulation. The simulation results show that our scheme achieves fast worm propagation at the cost of very low false positive and false negative rates, and they show that a small fraction of nodes are infected in the network under the presence of our scheme.

## CHAPTER 8

### CONCLUSION

In conclusion, we proposed a framework with five components to robustly and efficiently detect and prevent wide-spread node compromise in wireless sensor networks. Specifically, as the first component, zone-based reputation scheme quickly detects and revokes compromised nodes with little overhead. As the second component, mobile malicious node detection scheme quickly finds out mobile compromise nodes and isolates them from the network in fully distributed manner. As the third and the fourth components, static and mobile replica detection schemes achieve fast detection and revocation of static and mobile replica nodes with reasonable overhead. As the fifth component, worm propagation detection scheme efficiently and effectively detects and blocks worm propagation in fully distributed manner. Through analysis and experimental study, we demonstrated that these schemes achieve robust and efficient detection and prevention of wide-spread node compromise. In particular, static replica detection schemes achieve 100% replica detection and zero false positive with at least 66% less communication overhead than the prior work. Mobile malicious node detection scheme detects mobile node compromise with virtually zero overhead while sustaining false positives and false negatives below at most 1%. Both zoneTrust and mobile replica detection schemes detect the compromised nodes and replica nodes at the cost of  $O(1)$  message overhead with false positives and false negatives of at most 1%. Finally, worm propagation detection scheme detects worm propagation while refraining the fraction of the infected nodes below at most 12.7% and restricting false positives and false negatives below at most 1%.

## REFERENCES

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks : a survey. *Computer Networks* 38(4):393–422, March 2002
- [2] D. Boneh and M.K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, Pages:213–229, August 2001.
- [3] J-Y. L. Boudec and M. Vojnović. Perfect simulation and stationary of a class of mobility models. In *IEEE INFOCOM*, Pages:2743–2754, March 2005.
- [4] J.B. Broch, D.A. Maltz, D.B. Johnson, Y-C. Hu, and J.G. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *ACM MobiCom*, Pages:85–97, October 1998.
- [5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends, and Applications*, 2(5):483–502, 2002.
- [6] S. Čapkun and J.P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):221–232, February 2006.
- [7] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, Pages:197–213 , May 2003.
- [8] H. Chan, A. Perrig, and D. Song. Secure hierarchical in-network aggregation in sensor networks . In *ACM CCS*, Pages:278–287, October 2006.
- [9] Y. Chen, W. Trappe, and R.P. Martin. Attack detection in wireless localization. In *IEEE INFOCOM*, Pages:1964–1972, May 2007.

- [10] X. Cheng, F. Liu, and D. Chen. Insider attacker detection in wireless sensor networks. In *IEEE INFOCOM*, Pages:1937–1945, May 2007.
- [11] H. Choi, S. Zhu, and T.F La Porta. {SET}: detecting node clones in sensor networks. In *IEEE/CreateNet SecureComm*, Pages:341–350, September 2007.
- [12] C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA International Conference on Cryptography and Coding*, Pages:360–363, December 2001.
- [13] M. Conti, R.D. Pietro, L.V. Mancini, and A. Mei. A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks. In *ACM Mobihoc*, Pages:80–89, September 2007.
- [14] T.M. Cover and J.A. Thomas. Elements of information theory. *Wiley-Interscience*, 2006.
- [15] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme. Robomote: enabling mobility in sensor networks In *IEEE IPSN*, Pages:404–409, April 2005.
- [16] D.J. Daley and J. Gani. Epidemic modeling: an introduction. Cambridge University Press, 1999.
- [17] F. Delgosha and F. Fekri. Threshold key-establishment in distributed sensor networks using a multivariate scheme. In *IEEE INFOCOM*, Pages:1–12, April 2006.
- [18] J. Deng, R. Han, and S. Mishra. Security support for in-network processing in wireless sensor networks. In *ACM SASN*, Pages:83–93, October 2003.
- [19] J. Deng, R. Han, and S. Mishra. Defending against path-based DoS attacks in wireless sensor networks. In *ACM SASN*, Pages:89–96, November 2005.



- [20] D. Dong, M. Li, Y. Liu, and X.Y. Li. Topological detection on wormholes in wireless ad hoc and sensor networks. In *IEEE ICNP*, Pages:314–323, October 2009.
- [21] W. Du, J. Deng, Y. S. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *ACM CCS*, Pages:42–51, October 2003.
- [22] W. Du, J. Deng, Y.S. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *IEEE INFOCOM*, Pages:586–597, March 2004.
- [23] W. Du, L. Fang, and P. Ning. {LAD}: localization anomaly detection for wireless sensor networks. In *IEEE IPDPS*, Pages:874–886, April 2005.
- [24] W. Du, R. Wang, and P. Ning. An efficient scheme for authenticating public keys in sensor networks. In *ACM MobiHoc*, Pages:58–67, May 2005.
- [25] X. Du and Y. Xiao. Chapter 17: A survey on sensor network security *Springer Wireless Sensor Networks and Applications*, 2008
- [26] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *ACM CCS*, Pages:41–47, November 2002.
- [27] D.B. Faria and D.R. Cheriton. Radio Layer Security: Detecting identity-based attacks in wireless networks using signalprints. In *ACM WiSe*, Pages:43–52, September 2006.
- [28] A. Francillon and C. Castelluccia. Code Injection Attacks on Harvard-Architecture Devices. In *ACM CCS*, Pages:15–26, October 2008.
- [29] S. Ganeriwal, S. Čapkun, C.C. Han, and M.B. Srivastava. Secure time synchronization service for sensor networks. In *ACM WiSe*, Pages:97–106, September 2005.
- [30] S. Ganeriwal and M. Srivastava. Reputation-based framework for high integrity sensor networks. In *ACM SASN*, Pages:66–77, October 2004.

- [31] T. Goodspeed. Exploiting wireless sensor networks over 802.15.4. In *Texas Instruments Developer Conference*, 2008.
- [32] Q. Gu and R. Noorani. Towards self-propagate mal-packets in sensor networks. In *ACM WiSec*, Pages:172–182, April 2008.
- [33] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, and S. Eberle, H. Chang. Sizzle: a standards-based end-to-end security architecture for the embedded internet. In *IEEE PerCom*, Pages:247–256, March 2005.
- [34] C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: the need for secure systems. In *Technical Report CU-CS-990-05, Department of Computer Science, University of Colorado at Boulder*, January 2005.
- [35] J. Ho, D. Liu, M. Wright, and S.K. Das. Distributed detection of replica node attacks with group deployment knowledge in wireless sensor networks. In *Elsevier Ad Hoc Networks Journal*, 7(8):1476–1488, 2009.
- [36] J. Ho, M. Wright, and S.K. Das. Fast detection of replica node attacks in sensor networks using sequential analysis. In *IEEE INFOCOM*, Pages:1773–1781, April 2009.
- [37] J. Ho, M. Wright, and S.K. Das. ZoneTrust: fast zone-based node compromise detection and revocation in sensor networks using sequential analysis. In *IEEE SRDS*, Pages:217–223, September 2009.
- [38] L. Hu and D. Evans. Secure aggregation for wireless networks. In *Workshop on Security and Assurance in Ad Hoc Networks*, Pages:384–391, January 2003.
- [39] L. Hu and D. Evans. Localization for mobile sensor networks. In *ACM Mobicom*, Pages:45–57, September 2004.
- [40] X. Hu, T. Park, and K. G. Shin. Attack-tolerant time-synchronization in wireless sensor networks. In *IEEE INFOCOM*, Pages:41–45, April 2008.

- [41] Y.C. Hu, A. Perrig, and D.B. Johnson. Packet leases: A defense against worm-hole attacks in wireless ad hoc networks. In *IEEE INFOCOM*, Pages:1976–1986, April 2003.
- [42] J. Jung, V. Paxson, A.W. Berger, and H. Balakrishnan. Fast port scan detection using sequential hypothesis testing. In *IEEE Symposium on Security and Privacy*, Pages:211–225, May 2004.
- [43] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks Journal*, 1(2-3):293–315, September 2003.
- [44] L. Lazos, S. Čapkun, and R. Poovendran. {ROPE}: Robust position estimation in wireless sensor networks. In *IEEE IPSN*, Pages:324–331, April 2005.
- [45] F. Li and J. Wu. Mobility reduces uncertainty in {MANET}. In *IEEE INFOCOM*, Pages:1946–1954, May 2007.
- [46] Z. Li, W. Trappe, Y. Zhang, and B. Nath. Robust statistical methods for securing wireless localization in sensor networks. In *IEEE IPSN*, Pages:91–98, April 2005.
- [47] Z. Li, W. Xu, R. Miller, and W. Trappe. Securing wireless systems via low layer enforcements. In *ACM WiSe*, Pages:33–42, September 2006.
- [48] R. C. Linger. Systematic generation of stochastic diversity as an intrusion barrier in survivable systems software. Proc. of the Thirty-Second Annual Hawaii International Conference on Systems Sciences (HICSS), January 1999.
- [49] A. Liu and P. Ning. TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks. In *IEEE IPSN*, Pages:245–256, April 2008.
- [50] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *ACM CCS*, Pages:52–61, October 2003.
- [51] D. Liu, P. Ning, and W. Du. Attack-resistant location estimation in sensor networks. In *IEEE IPSN*, Pages:99–106, April 2005.

- [52] D. Liu, P. Ning, and W. Du. Group-based key pre-distribution in wireless sensor networks. In *ACM WiSe*, Pages:11–20, September 2005.
- [53] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in tinyOS based on elliptic curve cryptography. In *IEEE SECON*, Pages:71–80, October 2004.
- [54] S. Madria and J. Yin. SeRWA: a secure routing protocol against wormhole attacks in sensor networks. *Ad Hoc Networks Journal*, Article in Press.
- [55] T. Park and K. G. Shin. Soft tamper-proofing via program integrity verification in wireless sensor networks. In *IEEE Trans. Mob. Comput.*, 4(3):297–309, 2005.
- [56] S. PalChaudhuri, J-Y. L. Boudec, and M. Vojnović. Perfect simulations for random trip mobility models. In *38th Annual Simulation Symposium*, Pages:72–79, April 2005.
- [57] N. Patwari and S. K. Kasera. Robust location distinction using temporal link signatures. In *ACM MobiCom*, Pages:111–122, September 2007.
- [58] B. Parno, A. Perrig, and V.D. Gligor. Distributed detection of node replication attacks in sensor networks. In *IEEE Symposium on Security and Privacy*, Pages:49–63, May 2005.
- [59] B. Parno, M. Luk, E. Gaustad, and A. Perrig. Secure sensor network routing: a cleanslate approach. In *ACM CoNEXT*, December 2006.
- [60] B. Przydatek, D. Song, and A. Perrig. {SIA}: secure information aggregation in sensor networks. In *ACM SenSys*, Pages:69–102, November 2003.
- [61] J. Reich and E. Sklar. Robot Sensor Networks for Search and Rescue. In *IEEE International Workshop on Safety, Security and Rescue Robotics*, 2006.
- [62] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. {SWATT}: softWare-based attestation for embedded devices. In *IEEE Symposium on Security and Privacy*, Pages:272–282, May 2004.

- [63] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, Pages:47–53, August 1984.
- [64] M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim. Remote software-based attestation for wireless sensors. In *ESAS*, Pages:27–41, July 2005.
- [65] V. Shukla and D. Qiao. Distinguishing data transience from false injection in sensor networks. In *IEEE SECON*, Pages:41–50, June 2007.
- [66] H. Song, S. Zhu, and G. Cao. Attack-resilient time synchronization for wireless sensor networks. *Ad Hoc Networks*, 5(1):112–125, January 2007.
- [67] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou. TinySeRSync: secure and resilient time synchronization in wireless sensor networks. In *ACM CCS*, Pages:264–277, 2006.
- [68] 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*, Pages:1–13, April 2006.
- [69] D. Sy and L. Bao. {CAPTRA}: coordinated packet traceback. In *IEEE IPSN*, Pages:152–159, April 2006.
- [70] S. Tilak, V. Kolar, N.B. Abu-Ghazaleh, and K.D. Kang. Dynamic localization protocols for mobile sensor networks. In *IEEE MASS*, October 2004.
- [71] G. Theodorakopoulos and J.S. Baras. Game theoretic modeling of malicious users in collaborative networks. *IEEE Journal on Selected Areas in Communications*, 26(7):1317–1326, 2008.
- [72] A. Wald. Sequential analysis. *Dover Publications*, 2004.
- [73] H. Wang, B. Sheng, C.C. Tan, and Q. Li. Comparing symmetric-key and public-key based security schemes in sensor networks: a case study of user access control. In *IEEE ICDCS*, Pages:11–18, 2008.

- [74] R. Wang, W. Du, and P. Ning. Containing denial-of-service attacks in broadcast authentication in sensor networks. In *ACM MobiHoc*, Pages:71–79, September 2007.
- [75] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer* 35(10):54–62, 2002
- [76] M. Xie, H. Yin, and H. Wang. An effective defence against email spam laundering. In *ACM CCS*, Pages:179–190, October 2006.
- [77] K. Xing, F. Liu, X. Cheng, and H.C. Du. Real-time detection of clone attacks in wireless sensor networks. In *IEEE ICDCS*, Pages:3–10, June 2008.
- [78] Y. Yang, X. Wang, S. Zhu, and G. Cao. {SDAP}: a secure hop-by-hop data aggregation protocol for sensor networks. In *ACM MOBIHOC*, Pages:356–367, 2006.
- [79] Y. Yang, X. Wang, S. Zhu, and G. Cao. Distributed software-based attestation for node compromise detection in sensor networks. In *IEEE SRDS*, Pages:219–230, October 2007.
- [80] Y. Yang, S. Zhu, and G. Cao. Improving sensor network immunity under worm attacks: a software diversity approach. In *ACM MobiHoc*, Pages:149–158, May 2008.
- [81] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. In *IEEE INFOCOM*, Pages:2446–2457, 2004.
- [82] F. Ye, H. Yang, and Z. Liu. Catching moles in sensor networks. In *IEEE ICDCS*, June 2007.
- [83] J. Yi, S. Yang, and H. Cha. Multi-hop-based monte carlo localization for mobile sensor networks. In *IEEE SECON*, Pages:162–171, June 2007.
- [84] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, August 2008.

- [85] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *IEEE INFOCOM*, Pages:1312–1321, April 2003.
- [86] L. Yu and J. Li. Grouping-based resilient statistical en-route filtering for sensor networks. In *IEEE INFOCOM*, Pages:1782–1790, April 2009.
- [87] J. Zhang, M. Firooz, N. Patwari, and S. K. Kasera. Advancing wireless link signatures for location distinction. In *ACM MobiCom*, Pages:26–37, September 2008.
- [88] W. Zhang, M. Tran, S. Zhu, and G. Cao. A random perturbation-based scheme for pairwise key establishment in sensor networks. In *ACM Mobihoc*, Pages:90–99, September 2007.
- [89] Y. Zhang, J. Yang, L. Jin, and W. Li. Locating compromised sensor nodes through incremental hashing authentication. In *DCOSS*, June 2006.
- [90] B. Zhu, V.G.K. Addada, S. Setia, S. Jajodia, S. Roy. Efficient distributed detection of node replication attacks in sensor networks. In *ACSAC*, Pages:257–267, December 2007.
- [91] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved by hop-by-hop authentication scheme for filtering injected false data in sensor networks. In *IEEE Symposium on Security and Privacy*, Pages:259–271, May 2004.
- [92] C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and early warning for internet worms. In *ACM CCS*, Pages:190–199, October 2003.

## BIOGRAPHICAL STATEMENT

Jun-Won Ho was born in Seoul, South Korea. He received his B.S. degree in Computer Science from Yonsei University, Seoul, South Korea. He also received his M.S. degree in Electrical and Computer Engineering from University of Wisconsin at Madison, Madison, Wisconsin and his Ph.D. degree in Computer Science from University of Texas at Arlington, Arlington, Texas. His current research interest is in the security in wireless sensor networks.