4-2007

Lightweight Intrusion Detection: A Second Line of Defense for Unguarded Wireless Sensor Networks

Vijay Subhash Bhuse
Western Michigan University

LIGHTWEIGHT INTRUSION DETECTION: A SECOND LINE OF DEFENSE FOR UNGUARDED WIRELESS SENSOR NETWORKS

by

Vijay Subhash Bhuse

A Dissertation
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Doctor of Philosophy
Department of Computer Science

Western Michigan University
Kalamazoo, Michigan
April 2007

LIGHTWEIGHT INTRUSION DETECTION: A SECOND LINE OF DEFENSE FOR UNGUARDED WIRELESS SENSOR NETWORKS

Vijay Subhash Bhuse, Ph. D.

Western Michigan University, 2007

Wireless sensor networks (WSN) are formed of stationary nodes with stringent resources (in terms of battery power, processor speed, memory and radio range). They have specific communication and traffic patterns.

Making sensor networks secure is especially challenging because of the wireless medium and the fact that WSN is physically unguarded. The compromise of sensor nodes may lead to the loss of secret information and tampering of the software. Hence, intrusion detection techniques must be designed to detect at least some of the most dangerous attacks. Further, these techniques should be lightweight to suit resource constrained nature of WSN.

We focus on proposing lightweight detection techniques for most dangerous attacks such as masquerade, Sybil, packet dropping, sinkhole, data-forging by an aggregator, exhaustion, HELLO flood and infusing invalid information. We also propose techniques which add new nodes securely, allow sensor nodes to send anomalies or information about detected attacks/attackers to the base station and isolate detected attackers.

MG method for detecting masquerade/Sybil is based on overhearing the communication of the immediate neighbors. SRP method verifies the number of packets sent and received from nodes based on their *id*.

For periodic monitoring type of applications, we propose to detect packet dropping and sinkhole which estimates the number of packets a node should receive/send from/to its neighbors. Estimating the number of packets is possible because sensor nodes send data periodically to the base station using a deterministic traffic pattern. The proposed mechanism also detects exhaustion and HELLO flood attacks. Our technique (DPDSN) detects packet dropping paths and detects packet dropping nodes only if there is a need to do so.

We also propose overhearing based technique for detecting data-forging by sensor nodes and aggregator. Our work in detecting invalid source of information (IASN) is based on expecting certain kind of data from a certain neighbor.

We analyze the probability of success and overhead of these techniques. These solutions do not substitute cryptography based techniques which generally provide the first line of defense. Instead they compliment the first line of defense. These solutions are necessary because physical capture of a sensor node is easily possible.

UMI Number: 3263334

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Dedicated to my parents Subhash and Vimal Bhuse

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research advisor Dr. Ajay Gupta for his guidance, support and encouragement throughout the course of my Ph. D. I would like to thank other members of the committee Dr. Mohsen Guizani, Dr. Leszek Lilien and Dr. Aaron Striegel (University of Notre Dame) for their intellectual contributions, discussions and help. I would also like to thank Dr. Zijiang Yang and Dr. Ala Al-Alfuquah for their help.

I sincerely acknowledge the members of the CERIAS research group (Purdue University) and the members of the Multimedia Systems Laboratory (University of Illinois, Chicago) for their technical input.

Finally I thank my colleagues Zille Huma Kamal and Mark Terwilliger for their help and all the interesting discussions we had in the WiSE Lab.

Vijay Subhash Bhuse

TABLE OF CONTENTS

Table of Contents – Continued

CHAPTER

Table of Contents – Continued

CHAPTER

Table of Contents – Continued

CHAPTER

Table of Contents – Continued

CHAPTER

Table of Contents – Continued

LIST OF TABLES

LIST OF FIGURES

List of Figures - Continued

List of Figures - Continued

ABBREVIATIONS

WSN: Wireless Sensor Network

ADT: Anomaly Detection Table

IASN: Information authentication in WSN

DPDSN: Detection of Packet Dropping for Sensor Network

MG: Mutual Guarding

SRP: Send Receive Packets

CHAPTER 1

INTRODUCTION

1.1 WSN and Intrusion Detection

Wireless Sensor Networks (WSN) consist of small devices—called sensor nodes—with RF radio, processor, memory, battery and sensor hardware. One can precisely and deeply monitor the environment with widespread deployment of these devices. Sensor nodes are resource-constrained in terms of the radio range, processor speed, memory size and power. WSN follow specific communication patterns as discussed in [1]. Apart from this, sensor nodes are generally stationary. The traffic rate is very low and generally the traffic is periodic as well. There may be long idle periods during which sensor nodes turn off their radio to save energy consumed by idle listening. Recharging or replacing batteries is expensive and may not even be feasible in some situations. Therefore, WSN applications need to be extremely energy-aware.

WSN is mostly unguarded. Hence, capturing a node physically, altering its code and getting private information like cryptographic keys is easily possible for an attacker. Wireless medium is inherently broadcast in nature. This makes them more vulnerable to attacks. Attacks can disrupt the operation of WSN and can even defeat the purpose of their deployment. An adversary can launch DoS attacks without much effort (e.g. even without cracking keys used for cryptography-based solutions). To be practical for real-life WSN deployments, techniques for detecting attacks should be lightweight. It is important to find nodes that are posing attacks and isolate them because physical capture and subsequent loss of secret information is easily possible.

Masquerade and Sybil are the most dangerous attacks because attacker can hide and perform other attacks. Data forging can be dangerous because it defeats the purpose of WSN deployment and can even be harmful (e.g. WSN is deployed for detecting fire and compromised sensors prevent WSN from reporting fire). Packet-dropping, sinkhole and exhaustion deplete WSN nodes of energy. Therefore it is important to design techniques to detect these attacks as well.

The focus of the dissertation is on designing lightweight techniques that can *detect* the most dangerous attacks. The proposed techniques should take into account important WSN characteristics like specific communication patterns, periodic traffic, aggregation [1], coverage and connectivity [2].

1.2 Related Work

In literature the term *intrusion* means both intrusion by outsider and insider abuse. Kumar [3] has categorized intrusions into two types,

- *Misuse or Signature-based detection*: Intruder takes advantage of weaknesses in the system and finds out a way to get in. We can formally define these attack patterns. These attack patterns are called as signatures. Therefore if new adversary tries to use known attacks to intrude then he will be caught if his pattern of attack matches some signature.
- *Anomaly detection*: In this type of intrusion detection, normal user behavior is defined and the intrusion detection system looks for anything that is anomalous hence suspicious. Anomaly detection assumes that intrusion is a kind of anomalous activity. If it detects anomalous behavior, it can detect an intrusion.

One of the earliest works on intrusion detection is commonly considered to be the one reported by Anderson [4], which introduced the idea of doing anomaly detection by creating profiles of normal use and detecting deviations from those profiles. This idea was later formally presented by Denning [4] in what is considered to be the seminal paper for modern intrusion detection. For a review of intrusion detection in wireless ad hoc networks, we refer the reader to the paper by Mishra *et al.* [6]. Zhang and Lee [7] proposed architecture for a distributed and cooperative intrusion detection system for ad hoc networks based on statistical anomaly detection techniques. To paraphrase from [6] this article does not discuss the actual detection techniques. Bhargav *et al.* [8] proposed an intrusion detection and response model to enhance security in AODV [9].

Marti *et al.* [10] discussed two techniques that detect compromised nodes that agree to forward packets but fail to do so. The authors use *watchdog*s that identify misbehaving nodes and a *pathrater* that helps routing protocols avoid these nodes. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. The watchdog does this by listening promiscuously to the next node's broadcast transmissions. If the next node does not broadcast the packet, it is misbehaving and the watchdog detects it. Every time a node fails to forward a packet, the watchdog increments the failure-tally. If the tally exceeds a certain threshold, it is determined that the node is misbehaving; this node is then avoided with the help of the *pathrater*. The *pathrater* combines knowledge of misbehaving nodes with link reliability data to pick the route most likely to be reliable. Each node maintains a rating for every other node it knows about in the

network. It calculates a path metric by averaging the node ratings in the path. The overhead of passive continuous passive listening is formidable for WSNs.

Buchegger *et al.* [11] proposed a mechanism that detects misbehaving nodes by means of observations or reports about several types of attacks. This allows nodes to find routes around misbehaving nodes and to isolate them from the network. Nodes have a *monitor* for observations, *reputation records* for first-hand observations and trusted second-hand reports, *trust records* to control trust given to received warnings, and a *path manager* to adapt their behavior according to reputation of other nodes. This approach involves continuous monitoring similar to Marti's approach and collecting information about intrusion detections at other places in the network. The overhead is prohibitive for WSNs.

Michiardi *et al.* [12] proposed a collaborative reputation mechanism that has a *watchdog* component. However, it is complemented by a reputation mechanism that differentiates between subjective reputation (observations), indirect reputation (positive reports by others), and functional reputation (task specific behavior). They are weighted for a combined reputation value used to make decisions about cooperation with or gradual isolation of a node. This approach involves continuous monitoring and collecting information about intrusion detections at other places in the network for specific functions. The overhead is too high for WSNs.

Huang *et al.* [13] proposed a mechanism that needs separate monitoring nodes, specifically one monitor per cluster (nodes that are in one-hop range form a cluster). The approach requires monitors to be active. If there is one monitor per cluster, the monitor does most of the work. In WSNs, there is a risk that monitor nodes run out of

energy before the network does or before the network gets partitioned. This contradicts one of the main goals of prolonging WSN lifetime and keeping WSN connected as much as possible (since battery replacement is a very costly or unavailable alternative).

Demirbas *et al.* [14] present a solution for detecting Sybil attack on WSN. It is based on received signal strength indicator (RSSI) values. The proposed solution needs collaboration of one other node apart from the receiver. Authors show that even though RSSI is time-varying and unreliable and radio transmission is non-isotropic, using ratio of RSSIs from multiple receivers it is feasible to overcome these problems.

Ngai *et al.* [15] present an algorithm for detecting the intruder in a sinkhole attack on WSN. The algorithm first finds a list of suspected nodes, and then identifies the intruder in the list through a network flow graph. The algorithm also deals with cooperative malicious nodes that attempt to hide the real intruder.

Piro *et al.* [16] show that passively monitoring traffic in the network can detect a Sybil attacker that uses a number of network *id*s simultaneously. They show that this detection can be done by a single node, or that multiple trusted nodes can join to improve the accuracy of detection. They show that it is possible to differentiate between a single attacker spoofing many addresses and a group of nodes traveling in close proximity. The solution is for MANETs.

Watchdog's weaknesses [6] are that it might not detect a misbehaving node in the presence of

1. Collisions.

2. Limited transmission power: A misbehaving node could limit its transmission power such that the signal is strong enough to be overheard by the previous node but too weak to be received by the true recipient.

The Local Intrusion Detection System [17] is distributed and uses mobile agents on each of the nodes of the ad hoc network. A distributed IDS has been proposed [18] in which each node on the network has an IDS agent running on it. The IDS agents in the network collaborate to decide when and how the network is being attacked. The architecture is divided into parts: the mobile IDS agent, which resides on each node in the network, and the stationary secure database, which contains global signatures of known misuse attacks and stores patterns of each user's normal activity in a non-hostile environment [1].

Kachirski and Guha have proposed a distributed intrusion detection system for ad hoc wireless networks based on mobile agent technology [19]. By efficiently merging audit data from multiple network sensors, their bandwidth-conscious scheme analyzes the entire ad hoc wireless network for intrusions at multiple levels, tries to inhibit intrusion attempts, and provides a lightweight low-overhead mechanism based on the mobile agent concept [1]. The CONFIDANT protocol [20] detects misbehaving nodes by means of first-hand and trusted second-hand observations or reports about several types of attacks and avoids misbehaving nodes.

Yu *et al.* [48] proposed a lightweight security scheme for detecting selective forwarding attacks. The detection scheme uses a multi-hop acknowledgement technique to launch alarms by obtaining responses from intermediate nodes.

Banerjee *et al.* [49] propose an ant colony based intrusion detection mechanism which could also keep track of the intruder trials. The proposed technique could work in conjunction with the conventional machine learning based intrusion detection techniques to secure the sensor networks.

Agah *et al.* [50] proposed a protocol based on game theory which detects the presence of nodes that agree to forward packets but fail to do so.

Da Siva *et al.* [51] proposed a simple rule based IDS that detects wormhole, jamming and data alteration. They use cryptography and do not assume that physical capture of a node is possible.

1.3 Definitions of Attacks

Masquerade [52] is a type of attack in which one system entity illegitimately poses as (assumes the identity of) another entity. As paraphrased from [53] the forging of multiple identities is a Sybil attack [54] on the system.

A subverted sensor node [55] can simply neglect to forward certain or all packets. An attacker may also drop packets to or from certain victims, such as base stations or other servers. In a sinkhole attack [1], a malicious node uses the faults in a routing protocol to attract much traffic from a particular area, thus creating a sinkhole.

An attacker may be able to [55] perform a denial of service attack on the network by inducing repeated retransmission attempts. Even in the absence of high-rate traffic, if a node must continually retransmit due to collisions or have to route heavy traffic, eventually its energy may be exhausted.

In a HELLO flood attack [1] a malicious node can send, record or replay HELLO messages with high transmission power. It creates an illusion of being a neighbor to many nodes in the networks and can confuse the network routing badly.

We define data forging by an aggregator as an attack in which aggregator either forges the data sent by sensor nodes or the final result.

1.4 WSN Model

We assume that N sensor nodes equipped with isotropic antenna of range r and sensing radius r, are uniformly distributed in a square area of length W such that they completely cover the area and remain connected. The base station is placed in one corner of the square area. Clusterheads aggregate [30] sensor readings from sensors that are in their communication range and forward a single packet towards the base station. The aggregated data may be forwarded by sensor nodes or clusterheads. Clusterheads (also called as aggregators) are normal sensor nodes and the role of clusterhead is rotated among the nodes. We define *iteration* as the data gathering cycle during which each sensor node sends locally sensed data to the clusterhead and clusterheads forward the aggregated data to the base station. The base station is resource-rich whereas sensor nodes are resource-constrained. We assume that the sensor nodes are stationary. *PWR* denotes the initial battery power (energy) of sensor nodes.

1.5 Trust Model

We assume that the base station is physically guarded and cannot be compromised. Every sensor node shares a separate secret key with the base station. This secret key is used to encrypt the locally detected intrusion information or

anomalies which are sent to the base station. This secret key can be embedded in a sensor node, at design time, while it is programmed. Based on the anomalies or intrusion information from sensors, base station guesses about attacks and can initiate the appropriate action. This centralized approach is necessary because individual sensor nodes can be easily compromised. The base station securely informs about the addition of new nodes to their neighbors. Therefore it is safe to assume that nodes know their neighbors. Below we discuss the intrusion detection system model.

1.6 Intrusion Detection System Model

If each anomaly or intrusion noticed by a node is reported to the base station then it incurs an overhead that is proportional to the number of hops it takes a node to reach the base station. These messages are encrypted using symmetric cryptographic algorithm. Elimination of nodes that are posing attacks is the most difficult problem because nodes capable of performing masquerade or Sybil attack can join again with different *id*. Elimination messages are broadcast messages and they must be authenticated. For this purpose, we propose the use of broadcast authentication protocol, μTESLA [41]. Even the addition of new nodes is announced by the base station using μTESLA. One way function SHA1 can be used to compute message authentication code (MAC) of the message that is broadcasted by the base station. MAC is computed using a key chain that the base station computes. K_0 is a commitment to the key chain and it is safe to assume that all the nodes have this key K_0 from design time. Losing K_0 does not affect μTESLA. μTESLA also requires nodes to be loosely time synchronized.

1.7 Dissertation Statement

Intrusion detection techniques based on mutual guarding, statistical information about traffic and existing system information can be used to detect attacks such as masquerade, Sybil, packet-dropping, sinkhole, exhaustion, HELLO flood and data forging (by an aggregator) for WSN.

1.8 Contributions of the Dissertation

The main contributions of this dissertation are,

1. Identifying the most dangerous attacks and
2. Designing and evaluating lightweight solutions to detect these attacks.

1.9 Organization of the Dissertation

This dissertation is organized as follows. Chapter 2 identifies the most dangerous attacks by finding their precursors and studying their effects. Chapter 3 discusses the solutions for detecting masquerade attack. Chapter 4 describes the solutions for detecting Sybil. Chapter 5 describes solutions to detect packet dropping paths and nodes. Chapter 6 presents the solutions for detecting sinkhole, exhaustion and HELLO flood. Chapter 7 discusses solution to detect data forging by an aggregator. Chapter 8 discusses detection of invalid source of information. Chapter 9 concludes the dissertation.

CHAPTER 2

RESEARCH CHALLENGES IN INTRUSION DETECTION FOR WIRELESS SENSOR NETWORKS

Abstract

This chapter discusses two issues. First, we discuss characteristics and vulnerabilities of WSN that necessitate designing intrusion detection (ID) solutions specialized for WSN. Second, we identify research challenges in ID for WSN. We also identify a new type of attack on WSN, called phenomenon forging.

2.1 Introduction

Recall from section 1.1 that the first line of WSN defense, attack prevention, is not sufficient. The second line of defense, attack detection, using *intrusion detection* (*ID*) techniques is necessary. There are many publications (e.g., [21, 22]) about possible *intrusion detection systems* (*IDSs*) for a broader class of *ad hoc networks* (*AHNs*), which includes WSN. But unless the IDSs are based on efficient mechanisms for detection of a sufficient variety of attacks, we will be far away from practical ID solutions. To the best of our knowledge, the efficient ID mechanisms for detecting intrusions other than packet dropping and flooding attacks in AHNs remain to be investigated. This means that the existing solutions for AHNs do not cover a sufficiently broad scope of intrusions.

In the next section, we present other reasons why even these IDSs for AHNs, for the limited set of intrusions, cannot be directly imported for WSN. The most critical consideration in borrowing ID solutions from other AHNs is that energy resources in WSN are even more constrained. Energy is an expensive resource for WSN, because sensor nodes use batteries. Recharging or replacing batteries is expensive and,

depending on the deployment milieu, may even be impossible. To be practical for use in WSN, solutions for detecting intrusions should be lightweight.

The next section discusses characteristics and vulnerabilities of WSN that make general intrusion detection solutions for ad hoc networks infeasible for WSN. Section 2.3 identifies research challenges in the area of ID solutions for WSN. The first challenge is to identify the most dangerous attacks on WSN. We postulate that due to the limited WSN resources, ID in a WSN should be limited to only a few types of attacks—the ones most dangerous for WSN. Then, we outline other research challenges that indicate the importance of designing *lightweight ID mechanisms for WSN*.

2.2 Salient Features, Vulnerabilities and Controls for WSN

WSN can be viewed as a subcategory of wireless ad hoc networks. As such, the former must share some characteristics with the latter (like the use of the wireless medium) as well as have some distinguishing features. Salient features, listed below, make it difficult or impossible to simply import intrusion detection techniques for WSN from AHNs.

2.2.1 Salient WSN Features vs. Ad Hoc Network Features

The distinct characteristics of WSNin contrast to AHNs include the following:

1. Sensor nodes are more severely resource-constrained than AHNs. Uneven consumption of energy by sensor nodes is a bigger problem. Partitioning in a WSN is thus more probable, in effect seriously reducing the useful network lifetime.
2. Sensor nodes are mostly stationary.

3. Both the coverage of the area to be monitored by a WSN and the connectivity of its nodes must be taken into consideration during WSN deployment [2]. This is not an issue for AHNs in general.
4. Traffic patterns in WSN differ from traffic patterns in AHNs in at least the following ways:
   a) Unlike in AHNs, traffic patterns in WSN can be classified into three different categories [1]: many-to-one, one-to-many, or local communications. In *many-to-one communication*, many sensor nodes send readings to a base station or an aggregation point in the network. Typically, data is aggregated on its way to the base station to reduce the number of messages [1]. In *one-to-many communication*, a single node (typically a base station or an aggregator) floods several sensor nodes with a query or control information. Finally, in *local communication*, neighboring nodes send localized messages to discover each other and coordinate with each other.
   b) Traffic in WSN is not as randomly distributed as in AHNs. Since WSN are deployed to detect and report events to a base station, traffic is event-driven—which normally makes it bursty or periodic. Different routing protocols [23] and sleep-wakeup based MAC protocols [24] take into consideration this nature of WSN traffic.

2.2.2 WSN Vulnerabilities vs. AHN Vulnerabilities

WSNs are more vulnerable to attacks than AHNs due to the following reasons:

1. Sensor nodes are mostly physically unguarded. A capture of a single node by an attacker can result in a compromise of shared secrets or cryptographic keys.

2. Sensor nodes are more resource-constrained in terms of their radio range, processor speed, memory capacity and battery power.
3. DoS attacks can succeed more easily, since sensor nodes are resource-constrained. Thus, DoS attacks are more dangerous, more easily defying the purpose of WSN deployment, even without cracking cryptographic keys.
4. Due to specific traffic patterns (as discussed above), use of asymmetric cryptographic primitives incurs a heavy communication overhead [25]. As a consequence, asymmetric cryptography—which is orders of magnitude slower than the symmetric one—is infeasible for data aggregation, considering limited resources of sensor nodes.

2.2.3 Security Controls in WSN

Most common security controls used in all kinds of networks, AHNs included, are based on encryption. It is hard to imagine providing security controls for WSN without cryptographic solutions, but—due to resource limitations in WSN—these must be *lightweight* cryptographic solutions. Being lightweight, they will be even less effective than medium or heavyweight cryptographic solutions available for networks and AHNs, which are routinely complemented with ID systems.

Since lightweight encryption in WSN will allow even more successful exploits, ID solutions are even more important in WSN than in AHNs or other networks. At the same time, ID solutions for WSN are even more difficult to devise due to their severe resource constraints.

In view of these facts, we postulate to limit intrusion detection in a WSN to only a few types of attacks most dangerous for the WSN. Details follow.

2.3 Research Challenges in Lightweight Intrusion Detection for WSN

This section identifies the major challenges in developing lightweight intrusion detection techniques for WSN.

2.3.1 Finding the Most Dangerous Attacks

Simpler attacks on WSN can be precursors leading to more dangerous ones. Table 1 shows attack precursors for a number of common attacks. Detecting any of the precursors is a worthwhile goal as any simpler attack may precede more complicated and sophisticated attacks that are more difficult to detect. Finding precursors and detecting them as early as possible is also beneficial considering limitations on WSN resources.

Attacks	*Precursors*
Masquerades	Packet forging or Sybil attacks
Sybil attacks	Packet forging
Man-in-the-middle attacks	Packet forging and masquerades
False route requests	Packet forging and attacking routing protocols
Misdirections	Packet forging
Selective forwarding	Packet dropping
Sinkhole	Packet dropping

Table 2.1: Attacks and their precursors.

Major *effects* of some attacks are briefly enumerated in table 2. Masquerades and Sybil attacks do not do any direct harm, so they are not included in table 2. It should

also be noted that to create a sinkhole, adversary attracts traffic towards itself. An attacker exploits weaknesses in the routing protocol to launch this attack.

Attacks	*Effects*
Physical capture, Tampering, Jamming, Collisions, Unfairness	Unavailability
Man-in-the-middle, Packet dropping, Blackhole, Selective forwarding	Network partition, Exhaustion
Sinkhole, Flooding, False route request, Misdirection, Wormhole	Exhaustion
Selective forwarding	Unavailability, Exhaustion

Table 2.2: Attacks and their major effects.

The following parameters can be used to quantify threats posed by attacks:

1. An immediate threat vs. a long-term effect: Some attacks may pose immediate threats to WSN operations whereas some may not. The latter might still be very harmful in a long term if undetected.
2. Active vs. passive: Attacks may be active (e.g., packet forging) or passive (e.g., overhearing by adversary).
3. Amount of resources used by attackers: Some attacks may need quite resourceful attackers (e.g., HELLO flood attacks) or more than one attacker (e.g., DDoS), whereas others can be mounted even by nodes with just normal resources (e.g., blackholes).

4. WSN participation: Some attacks are possible only if an adversary joins the WSN prior to attacking it.

It might be hypothesized that a higher priority in detection should be given to immediate threats, active attacks, attacks that need lesser resources, and attacks that do not need WSN participation. (As a consequence, attacks that do not pose immediate threats, passive attacks, attacks that require more resources, and the attacks that require WSN participation would be detected only as time and resources permit.)

Attacks leading to exhaustion do not pose an immediate threat but if undetected can decrease the network lifetime considerably. Since they do not pose an immediate threat, should their detection be done as time and resources permit? The attacks caused by packet forging lead to more dangerous and sophisticated attacks like man-in-the-middle. Attacks caused by packet dropping lead mainly to exhaustion of energy. But if a malicious node drops packets continuously, then its neighbors might wrongly conclude that it is dead. This may lead to network partitions.

2.3.2 Identifying Properties that Watchdogs can Monitor

Watchdogs have been proposed mainly for detecting packet dropping attacks [10, 12]. We propose using watchdogs to detect masquerade attacks. It is very important to identify the minimum amount of data, information and properties that watchdogs need to monitor to be able to detect a given number of attacks.

By monitoring packets, watchdogs can extract the following information and use it for attack detection:

1. The received signal strength indicator (RSSI) value for the signal, time of flight for a bi-directional communication, time of transmission or reception for a packet, or other physical, temporal and spatial properties of the received packet.
2. Nodes receiving a given packet or at least the network area that receives the packet.
3. Application-specific information that can be used to detect intrusions. Examples are a timestamp or a location of the network area where a query needs to be performed.

Design of watchdog-based detection mechanisms should consider the following important issues and criteria:

1. Watchdogs may need to observe a certain minimum number of packets to detect an attack.
2. Some watchdogs may need continuous monitoring whereas others may need periodic monitoring. The latter are preferable for resource-constrained WSN nodes.
3. Watchdogs may monitor the behavior of nodes, paths or clusters. Watchdogs that can detect attacks by just observing end-to-end behavior of a path are preferable over watchdogs that need to monitor every single node.
4. Current watchdog-based mechanisms [10] assume that the area in which a signal can be received is circular, with the transmitting node at the center of the circle. In practice the area is not circular. Watchdog-based mechanisms should adapt to the actual shapes of radio ranges.

5. Some detection techniques may require collaboration among watchdogs [12]. In such cases, providing a secure channel of communication between watchdogs is necessary. It is more difficult in the presence of attackers.
6. Collisions and hidden nodes pose problems to watchdog-based mechanisms and result in a large number of false alarms and misses [10].

2.3.3 Informing the Network about Locally Detected Intrusions

Even though intrusions are detected at nodes running host-based IDSs, informing either the whole network or a part of the network about the intrusion or quarantining the misbehaving nodes is a challenge because of the following reasons:

1. It is costly to provide a secure channel of communication for a resource-constrained WSN. This is the case even with lightweight cryptographic algorithms having low computational intensity.
2. Capture of a single sensor node results in compromising a shared cryptographic key. This is much more probable in WSN than in AHNs in general since sensor nodes are typically physically unguarded.
3. The adversary can eavesdrop on the wireless medium and can extract and misuse information shared between watchdogs. (Lightweight encryption can help.)

One more question remains open. If a secure communication channel cannot be provided, how can local ID information be shared among sensor nodes in such a way that adversary gets either no or only a part of the useful information? (And, in the latter case, cannot use it to pose any further attacks on the network?)

The number of adversaries present in the area and their locations will have a significant impact on the solutions.

2.3.4 Identifying DoS Attacks Specific to WSN

WSNs are especially vulnerable to DoS attacks (e.g. exploiting buffer overflows) because WSN nodes are resource-constrained. Security primitives based on cryptography are not sufficient to guard against DoS attacks because some DoS attacks can defeat the goal of the WSN even without cracking its keys. As an example, consider phenomenon forging defined below. Apart from that, DoS attacks can target proposed MAC and routing protocols for WSN.

***Phenomenon Forging*:** Suppose that a WSN is deployed to detect wildfires. Upon receiving an alert (which may contain a detection of a wildfire and its location), the response mechanism to extinguish wildfire gears up. An adversary can defeat the goal of the WSN by fooling many sensors with small "deceptive" fires (depending on the WSN intelligence, each could be just a lit match). In this way, the adversary can confuse the WSN with false alarms and exhaust WSN resources as well as the resources of the response mechanism. If a real wildfire starts after resource exhaustion, the WSN and the response mechanism might be unable to adequately respond. We call this attack *phenomenon forging*. It is specific to WSN, and it can be launched without cracking cryptographic keys or forging even a single data packet (the packets are all real – only the phenomenon is not).

2.3.5 Detecting and Identifying Resources Employed by an Adversary

Detecting strength of an attacker is important because it can affect the reaction of the response mechanism. The resources employed by an adversary can be measured by:

1. Counting the number of attackers.

2. Investigating resourcefulness of each attacker.
3. Investigating the way attackers communicate with each other. Well-connected attackers are more powerful.

Launching some attacks requires more effort. For example, an adversary wishing to create a sinkhole needs to participate in routing and find a loophole in it. A packet-dropping attack requires less effort.

2.3.6 Detection of Capture and Code Tampering

Code tampering is very difficult to prevent without a special hardware (incl. a processor) and a compiler [27]. WSN nodes are envisioned to become cheaper and smaller, eventually dust-sized [28]. They will be deployed in millions. To keep the costs low, it may not be possible to provide special hardware capabilities for such numerous and small nodes. This makes prevention of tampering difficult.

Since WSN nodes are physically unguarded, a physical capture is easy. Capture of a node can compromise shared secrets and keys.

2.4 Summary

Results of this chapter are published in [58]. Our contributions to identifying research challenges in intrusion detection for WSN can be summarized as follows:

1. We identified the specific properties of WSN that separate them from ad hoc networks and make it difficult to directly import intrusion detection solutions from ad hoc networks.
2. We proposed techniques to rank threats posed by attacks on WSN. We described the relationships between attacks and their precursors, and the effects of attacks on WSN.

3. We proposed using watchdogs for detection of masquerades and packet dropping attacks. We identified information that watchdogs can obtain.
4. We indicated the importance of securely informing the whole network or a part of it about locally detected intrusions.
5. We identified a new type of DoS attack, named *phenomenon forging*, which is specific to WSN.

CHAPTER 3

DETECTION OF MASQUERADE ATTACKS ON WIRELESS SENSOR NETWORKS

Abstract

We propose two lightweight techniques to detect masquerade attacks on wireless sensor networks (WSN). Our solutions take into consideration, important WSN properties like coverage, connectivity, data aggregation and specific communication patterns. The two proposed techniques complement each other when used concurrently. The mutual guarding (MG) technique does not work when nodes are not completely covered by their neighbors or when adversary has shorter transmission range than that of the sensor nodes. It also does not protect nodes near the boundary. Another technique based on the number of packets received and transmitted (SRP) does not have these drawbacks but is more complex. In this chapter, we present our proposed techniques and analyze their performance in terms of successful masquerade detection rate and traffic and computational overhead.

3.1 Introduction

Masquerade attacks can be very dangerous because adversaries can launch other attacks and can still hide and project themselves as legitimate nodes. Therefore, masquerade detection mechanisms are necessary. To be practical for real-life WSN deployments, techniques for detecting masquerade attacks should be lightweight.

We consider a setting in which an adversary is added to the network and it assumes the *id* of one of the nodes from *1* to *N*. There is no deterrent to prevent

adversaries from posing as one of the nodes with *id* from *1* to *N*. Adversaries must follow the MAC protocol being used by nodes in the network.

In the following sections we present our proposed detection strategies, analyze their performance, draw conclusions and summarize our findings.

3.2 Proposed Detection Techniques

We propose two techniques to detect masquerade attacks in WSN. Both techniques compliment each other when used concurrently.

3.2.1 MG: Mutual Guarding

As stated earlier, nodes are stationary and new nodes are added securely to the network. An attacker can assume the *id* of only the immediate neighbors because receiving a packet from someone that is not a neighbor is an anomaly. Similarly receiving a packet with source *id* same as your own *id* is also an anomaly. When two nodes *s* and *d* are in communication range, the common area (area with stripes as shown in figure 3.1) in which the packets sent by both of them can be received is said to be *mutually guarded* by them. In figure 3.1, when an adversary *A* sends a packet to *d* by setting source *id* to *s*, *s* also receives the packet. *s* detects the presence of attacker that masquerades as itself. Thus the adversary cannot masquerade as *s* or *d* without getting detected when it is located in the common area. Generalizing, when node *s* has neighbors around it and if the neighbors' transmission area overlaps with the whole area in which node *s* can transmit then the attacker cannot masquerade as any neighbor to node *s*. Receiving a packet sent by *A* by changing source *id* to *s*, is an anomaly for some node that has never received a packet from *s*. A node can thus detect the presence of an attacker that masquerades as *s* from these observations.

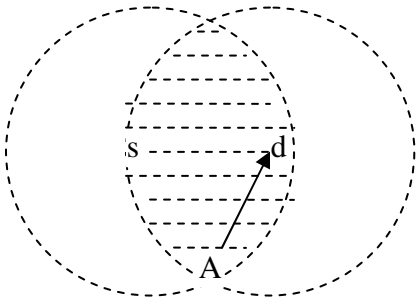


Figure 3.1: *s* overhears *A* masquerading as *s*.

The above proposed MG method detects the presence of attackers that have isotropic antennae with transmission range equal to or greater than that of the sensor nodes. The transmission area of node *s* needs to be completely guarded by its neighbors. WSN deployment takes both coverage and connectivity into account and hence MG method works for all the internal nodes (because they are completely guarded by neighbors). Note that if *A* has a directional antenna or a shorter range to reach *d* but not *s* then it will go undetected even if it is located in the common area. Our next method called SRP, is more complex but does not have this drawback.

3.2.2 SRP: Verification of the Number of Packets Sent and Received for Masquerade Detection

As stated earlier, if attacker assumes the *id* of a node that is not a neighbor, then it is an anomaly and attacker can be detected easily. Our proposed solution works for MAC protocols that avoid collisions by guaranteeing exclusive access to the RF channel at any given time (e.g. TDMA, 802.11 or CSMA/CA [31] with RTS, CTS, DATA and ACK). Using RF channel random access techniques, adversaries can masquerade the *id* of node *s* by transmitting data in the time slot allocated to *s*. If

adversary fails to follow MAC schedule or if collision is detected (for the above collision avoiding MAC protocols) then it is an anomaly. It indicates the presence of an attacker. Our proposed technique can now be described in detail as follows:

Let d be a node. Let s_i denote its i^{th} neighbor for $i > 0$. The following test is performed every T iterations. s_i keeps track of distinct number of packets sent (S_{sid}) to d during the time period that lasts for T iterations. Then d broadcasts a single packet containing the number of packets it received from its neighbors (R_{s1d}, R_{s2d}, R_{s3d}, …). If $R_{sid} > S_{sid}$ then we conclude that there is an adversary (one or more) that masquerades as s_i. Note that we assume for simplicity of discussion that the link layer is reliable which implies that packet losses due to noise, collisions etc. are handled reliably and a packet sent is received (albeit maybe after retransmits).

Attacker can perform DoS attack on the above solution but it can be detected easily. If adversary broadcasts a packet that d broadcasts then receivers will receive two such packets (one from adversary and one from d) in a time period that lasts for T iterations. This is an anomaly and it can be guessed that adversary is performing DoS attack on SRP.

3.3 Analysis

In this section, we analyze our proposed strategies in terms of success rate of detection, overhead and its effect on the network lifetime.

3.3.1 MG Method

We consider different scenarios as explained next.

3.3.1.1 Two Nodes Guard Each Other

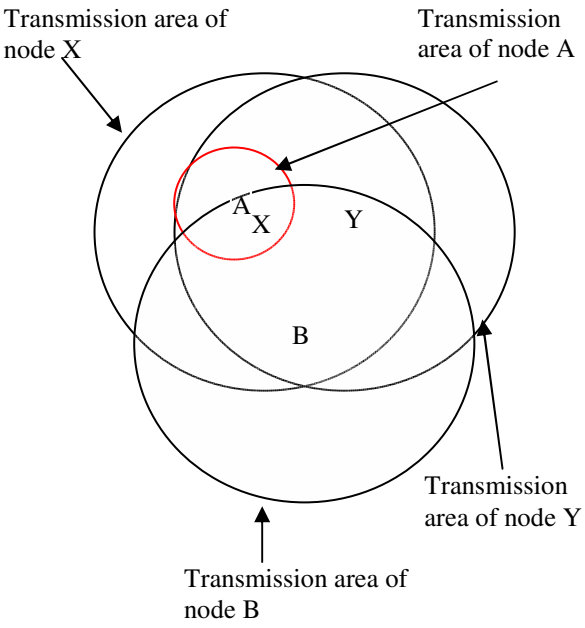


Figure 3.2: Two nodes *X* and *Y* that are distance *r* apart have transmission range of *R*.

Two nodes *X* and *Y* (of communication range *R*) have discovered that they are neighbors and hence they can receive packets from each other. An adversary of range *R* has to be inside the area occupied by two circles to communicate with either node (see figure 3.2 where node *B* is such an adversary). If an adversary sends a packet to *Y* with source field set to *X* from the common area occupied by 2 circles, then *X* will receive that packet as well. *X* can thus detect that someone is trying to masquerade as him. If adversary is in the common area occupied by two circles then masquerade can be detected. However note that we cannot detect an adversary (even if it sends from the common area), if it has a very small range because it can go very close to one node and send a packet (it is like whispering to someone so that others in the room do not hear anything. In figure 3.2 node A can be such an adversary). Another node will not be able to listen in on the transmission and hence will not receive the packet. Let

d be the distance between X and Y. The common area is $2R^2$ ($\cos^{-1}(d/2R) - (d/2R)(1-(d/2R)^2)^{0.5}$). Total area occupied by two circles is $2\Pi R^2 - 2R^2$ ($\cos^{-1}(d/2R) + (d/2R)(1-(d/2R)^2)^{0.5}$). If an adversary chooses the location randomly then the adversary is equally likely to be anywhere in the area occupied by the two circles. Therefore the probability of detection is $2R^2$ ($\cos^{-1}(d/2R) - (d/2R)(1-(d/2R)^2)^{0.5}$) / ($2\Pi R^2 - 2R^2$ ($\cos^{-1}(d/2R) + (d/2R)(1-(d/2R)^2)^{0.5}$). Ratio d/R can take values form 0 to 1. Figure 3.3 shows the probability of detection against the ratio d/R. If two nodes are very close to each other, the probability of detection is very high. Similarly if they are far apart the probability of detection becomes very low. This also gives insights on how to place nodes in a WSN. Adversary of range greater than R may go undetected because it can position itself such that it is able to communicate with only X or Y. In the next subsection, we show how the above idea can be extended for three nodes.

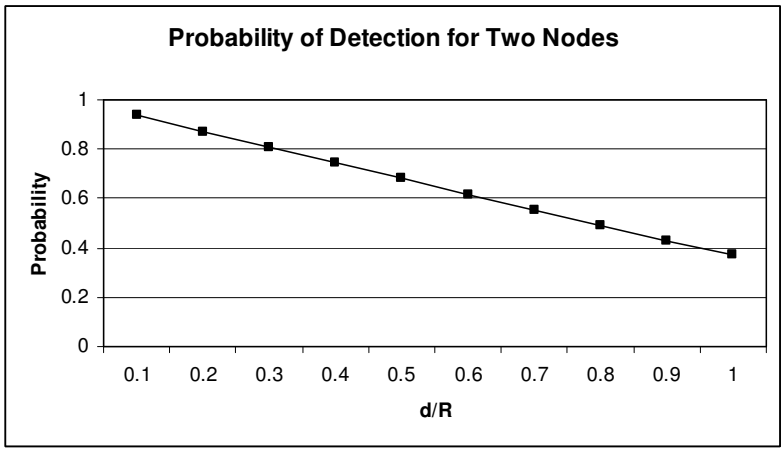


Figure 3.3: Probability of detection against d/R.

3.3.1.2 Three Nodes Guard Each Other

If three nodes are in the communication range of each other then they can guard each other from a masquerade attack better (see figure 3.4). For example if an adversary tries to send a packet to *B* by changing the source field to *A* from an area *AB* then *A* will receive that packet as well and can detect an intrusion. But adversary can masquerade as *C* from some part of an area *AB*. If an adversary sends a packet from area *AB*, area *BC* or area *AC*, it will be received by at least two nodes. Those nodes can guard each other. Similar to 2 nodes case, adversary of very small communication range will go undetected. Sometimes we cannot detect adversary of range greater than *R* because it can position itself in such a way that the packets it sends are received by only one node.

3.3.1.3 Detection of Masquerade Using Triangulation

The techniques discussed above can be extended for 4 nodes. The triangulation technique for detecting masquerade can be used as shown in figure 3.4. Let *R* be the radio range of all the nodes. Suppose node *X* has discovered that *A*, *B* and *C* are its neighbors. *A*, *B* and *C* are placed in such a way that the area in which they can transmit, completely occupy the area in which *X* can transmit. So if adversary has range less than or equal to *R* and it wants to send a packet to *X*, it must be in the inner circle.

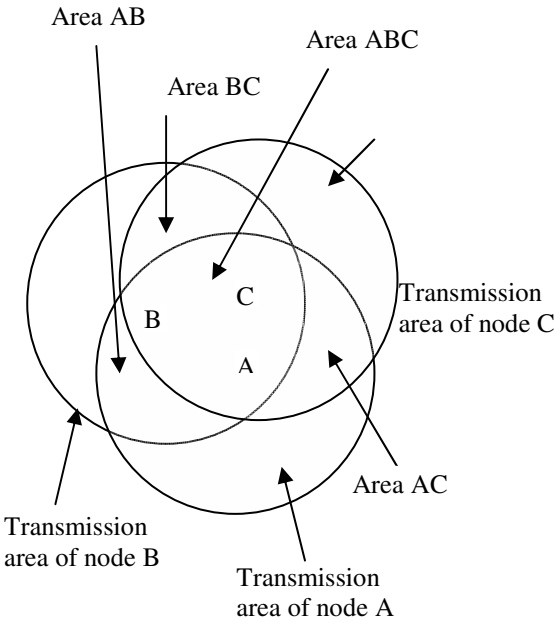


Figure 3.4: Three nodes *A*, *B* and *C*.

One or two neighbors of *X* will receive that packet. *A*, *B* and *C* have neighbor information and they know that they are not reachable from each other. So if *A* gets a packet with source field set to *B* or *C* then *A* will guess that it is a forged packet. If adversary tries to masquerade as *A* from area-*A* then *A* will receive that packet and will guess that there is an intrusion and someone is trying to masquerade as him. In some cases 2 nodes will detect masquerade. If adversary tries to send from area-*AC*, node *A* and *C* can detect it, similarly for area-*BC* and area-*AB*. Total area occupied by 4 nodes is $(\Pi + 3^{3/2})*R^2$. The area where packets from the adversary will be detected is ΠR^2. Therefore the probability of detection is $\Pi / (\Pi + 3^{3/2}) = 0.3768$. Probability of false positives is 0 and the probability of false negatives=0.6232.

3.3.1.4 Placement of Nodes

The above idea can be extended for a whole network. If we place nodes in such a way that the radio range of inner nodes is completely covered by surrounding nodes, then inner nodes are secured against masquerade attack. The nodes on the boundary are not entirely covered, so they are not completely guarded. Let N be the number of nodes. Let n_i be the set of neighbors any node i has for $1 \leq i \leq N$. If node i receives any packet with source field set to x such that $x \notin n_i$ then node i can drop that packet. If a node receives a packet with source field set to its own *id* then it can conclude that someone is trying to masquerade as him. If any node drops a packet that does not come from its immediate neighbor then a node can prevent masquerade. If there is a mechanism to securely inform other nodes about the intrusion then the attempts of masquerade can be prevented.

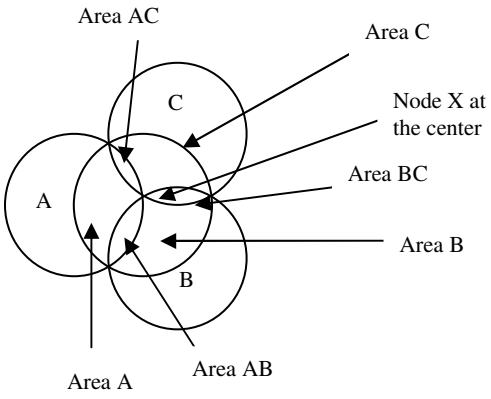


Figure 3.5: 4 nodes with transmission range R.

3.3.1.5 MG for the Whole Network

Consider sensor nodes deployed (as shown in figure 3.6) in a square area of length W. This deployment uses the minimum number of nodes to cover the whole

area. The deployment shown here is similar to the one presented in [2]. We consider this deployment only for analyzing MG method.

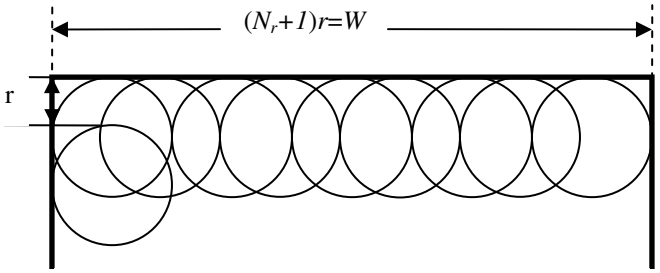


Figure 3.6: Optimal deployment for coverage and connectivity. Nodes are located at the center of the circles that are *r* apart.

The number of nodes in each row is N_r. Then $N= N_r^2$ and $W=(N_r +1)r$. Therefore $N_r = \left\lceil \frac{W}{r} \right\rceil - 1$. Nodes closest to the border are not completely covered. Hence they will not always be able to detect an attacker masquerading as their neighbor.

Theorem 3.1: If an attacker has a transmission range at least as large as the deployed sensor nodes, then for the optimal deployment as shown in figure 3.6 attacker can assume the *id*s of approximately $8N_r - 16$ nodes when it finds proper place to transmit from and it can perform masquerade attack on approximately $4N_r - 4$ nodes.

Proof: Only the nodes that are closest to the border are not completely covered. In figure 3.7, adversary *A* can assume the *id*s of *m*, *n*, or *y*. *x* will not be able to detect this masquerade attack because *m*, *n* and *y* will not receive the packets sent by *A*.

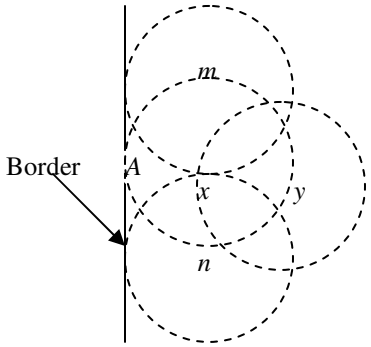


Figure 3.7: *x* will not detect adversary *A* masquerading as *m, n*, or *y*.

So adversary *A* can assume the *id*s of nodes that are 1 or 2 hops away from the border if it places itself at an appropriate position while performing the masquerade attack.

The number of nodes that are closest to the border is $4N_r - 4$ and only these nodes are vulnerable to the masquerade attack. Inner nodes are completely covered by their neighbors. The number of nodes that are 2 hops away from the border is $4(N_r - 2) - 4 = 4N_r - 12$. The total number of nodes that are not more than 2 hops away from the border is $8N_r - 16$. An adversary can assume the *id* of any of these nodes while performing a masquerade attack.■

3.3.2 SRP Method

In order to make it difficult for an adversary to guess the transmission time of a node, we can easily assume that nodes transmit packets at a random time during their allotted time slot. Collisions can be detected when a packet with the stronger signal is received last [32]. With minor modification to the packet structure in figure 3.8 (including source address in the tail), collisions in which the packet with the stronger signal is received first can also be detected [32]. This increases the collision detection rate to a theoretical maximum of nearly 100%. But when two packets arrive at exactly the same time, a collision cannot be detected at all [32].

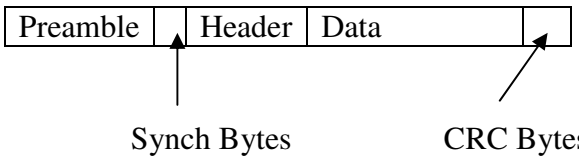


Figure 3.8: Packet format (Courtesy [32]).

There are two possibilities in this case. (i) Both packets are lost or (ii) One with stronger signal is received.

An adversary can transmit only in the time slot allocated to the node it is trying to masquerade as. If it transmits in the time slot that is not allocated to the node it is masquerading as, then it is an anomaly. An adversary can be easily detected. Let p_1 be the probability with which the packet sent by the adversary overlaps with the packet sent by the node that the adversary is trying to masquerade as. Let p_2 be the probability with which the adversary sends data packet at the same time as s and collision is detected at the receiver. Let p_3 be the probability with which both packets are lost provided collision cannot be detected. Let q be the probability with which adversary *guesses* that the previous packet was lost at the receiver because of interference. It can be noted that it is hard to guess this.

Theorem 3.2: The probability with which the adversary successfully masquerades one packet is $p_1.(1-p_2).(1-p_3) + p_1(1-p_2).p_3.q.(1-p_1)$. The probability that the adversary is detected while masquerading m packets is $1-(p_1.(1-p_2).(1-p_3) + p_1(1-p_2).p_3.q.(1-p_1))^m$.

Proof: Adversary successfully masquerades a packet in 2 ways.

In the first case, an adversary succeeds in masquerading a packet when (i) the packet sent by the adversary overlaps with the one sent by the original sender, (ii) collision cannot be detected at the receiver and (iii) an adversary has sent packet with the stronger signal. In this case receiver recovered a packet with stronger signal and it was sent by the adversary. Since a collision is not detected, the receiver increments R_{sid}. The original sender does not know that instead of its packet the one from the adversary with stronger signal is accepted. And it increments S_{sid}. When R_{sid}= S_{sid}, SRP is defeated. Therefore probability with which adversary succeeds is

p1(1-p2)(1-p3) (1)

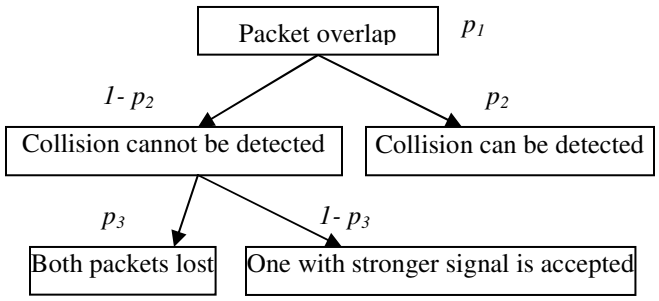


Figure 3.9: Probabilities in Lemma 3.2.

In the second case, an adversary successfully masquerades a packet when (i) there is a packet overlap, (ii) collision cannot be detected, (iii) both packets are lost, (iv) adversary guesses that both packets are lost and (v) adversary sends next packet such that it does not overlap with packets from s.

In this case both (one from the adversary and one from original sender) packets are lost at the receiver and collision could not be detected. Hence original sender does not know that its packet was lost in a collision and the receiver could not detect the collision. The original sender increments S_{sid}. An adversary sends the next packet and

the receiver increments its R_{sid} after receiving it. Then $R_{sid}= S_{sid}$, and SRP is defeated. The probability with which an adversary succeeds is

$$p_1(1-p_2).p_3.q.(1-p_1) \qquad (2)$$

Thus, the probability with which the adversary succeeds in masquerading a packet is

$$P= p_1.(1-p_2).(1-p_3) + p_1(1-p_2).p_3.q.(1-p_1) \qquad \text{from (1) and (2)}$$

The probability with which the adversary succeeds in masquerading m packets is P^m. The probability with which adversary is detected while masquerading m packets is $1-P^m$ and the theorem follows.■

For overhead analysis of SRP, we divide the square area into eccentric circular strips of width R with the base station at the center. This helps us count the approximate number of hops for clusterheads located in the strip to reach the base station. Let $W=cR$ for some constant c. We call an area "k-th strip" if all the nodes in that area are not farther than kR and not closer than $(k-1)R$ for $k \geq 1$. So nodes in strip k are not less than k hops away from the base station. We compute an area of strip k, S_k, next. From S_k we compute the approximate number of nodes, N_k, and the approximate number of cluster heads, C_k in strip k. Then we compute the lifetime of the WSN and the decrease in lifetime due to SRP.

Lemma 3.3: Area of strip k,

$$S_k=\frac{\pi(2k-1)R^2}{4} \qquad \text{for } 0<k\leq c$$

$$=\frac{(2k-1)R^2\theta}{2}-2A_1-2A_2 \qquad \text{for } k > c$$

where $\theta=\frac{\pi}{2}-2\cos^{-1}\left(\frac{W}{(k-1)R}\right)$.

$$\alpha = \cos^{-1}\left(\frac{W}{kR}\right) - \cos^{-1}\left(\frac{W}{(k-1)R}\right).$$

$$A_1 = \frac{k^2R^2\alpha}{2} - k^2R^2 \sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right).$$

$$A_2 = \left(\frac{-W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k}\right)\right)} + \frac{W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k-1}\right)\right)} \right) . \frac{R}{2} \sin\left(\frac{\pi}{4} + \frac{\theta}{2}\right).$$

θ, α, A_1 and A_2 are as shown in figure 3.10(a).

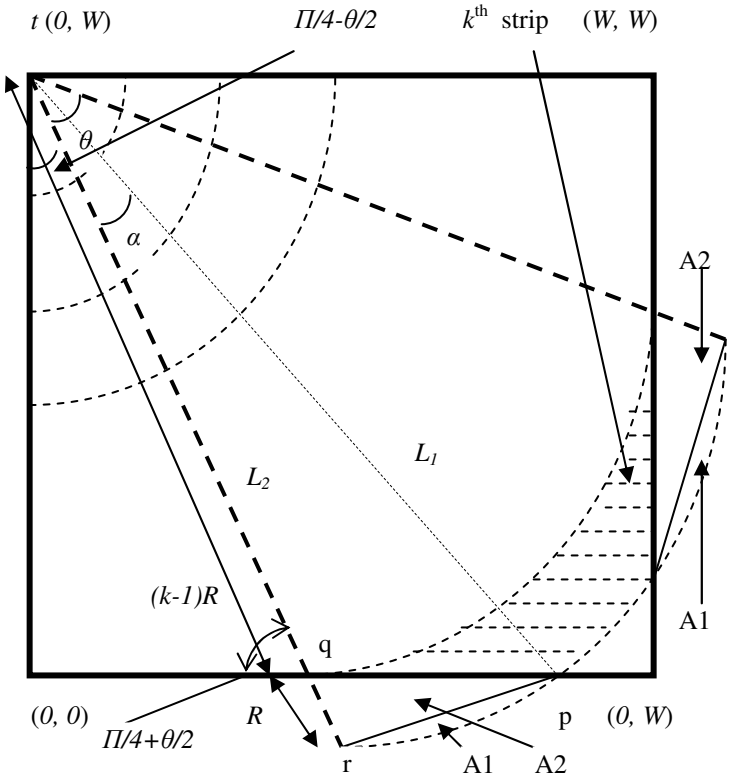


Figure 3.10(a): Area divided into c strips; S_k is marked with stripes.

Proof: An area of the sector with radius kR and angle θ is $\frac{k^2R^2\theta}{2}$. Difference between the areas of sector k and (k-1) gives us the area of the $k^{'th}$ strip for $0<k\leq c$.

$$S_k = \frac{\pi(2k-1)R^2}{4} \tag{3}$$

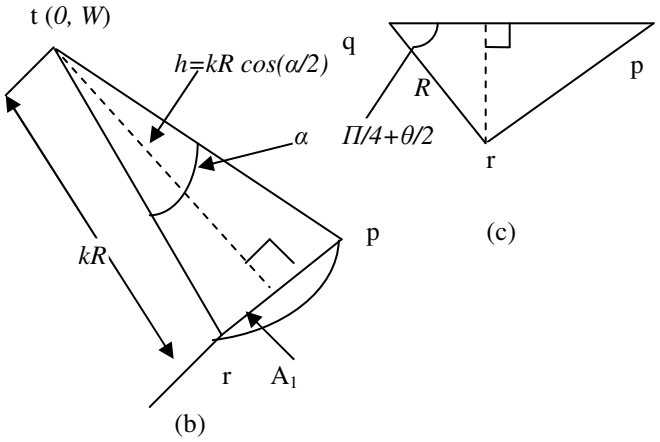


Figure 3.10: (b) Area A1 is shown above; (c) A2 is triangle pqr.

Next we calculate S_k strip for $k>c$. To calculate S_k we find the area of the track of width R, radius kR, angle θ with the center and subtract twice the addition of areas A_1 and A_2.

Next we calculate area A_1. By using trigonometry it can be shown that $\alpha = \cos^{-1}\left(\frac{W}{kR}\right) - \cos^{-1}\left(\frac{W}{(k-1)R}\right)$. To calculate A_1, we substract the area of the triangle tpr from the area of the sector shown in figure 3.10(b). By trigonometry, the height of the triangle, h is $\mathrm{kRcos}\left(\frac{\alpha}{2}\right)$ and length (p, r) = $2\mathrm{kRsin}\left(\frac{\alpha}{2}\right)$. Hence an area of the triangle tpr=0.5 * h * length (p, r) = $k^2R^2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right)$. An area of the sector = $\frac{k^2R^2\alpha}{2}$.

Therefore

$$A_1 = \frac{k^2R^2\alpha}{2} - k^2R^2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right) \tag{4}$$

We calculate A_2 next. To find the area of the triangle above we find the distance between points p and q and the height of the triangle (shown by dotted line). By

trigonometry, it can be shown that $\theta = \frac{\pi}{2} - 2\cos^{-1}\left(\frac{W}{(k-1)R}\right)$. We find the coordinates of points p and q. Slope of L_1 is $\tan\left(\pi - \sin^{-1}\left(\frac{c}{k}\right)\right)$. Hence an equation of line L_1 is $y = \tan\left(\pi - \sin^{-1}\left(\frac{c}{k}\right)\right)x + W$. X-intercept of L_1 is point p $\left(\frac{-W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k}\right)\right)}, 0\right)$. Similarly an equation of line L_2 is $y = \tan\left(\pi - \sin^{-1}\left(\frac{c}{k-1}\right)\right)x + W$. X-intercept of L_2 is point q $\left(\frac{-W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k-1}\right)\right)}, 0\right)$. The length of side pq, l(p,q)= $\left(\frac{-W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k}\right)\right)} + \frac{W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k-1}\right)\right)}\right)$ and the height of the triangle pqr is $R\sin\left(\frac{\pi}{4} + \frac{\theta}{2}\right)$. Hence an area of triangle pqr,

$$A_2 = \left(\frac{-W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k}\right)\right)} + \frac{W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k-1}\right)\right)}\right).\frac{R}{2}\ \sin\left(\frac{\pi}{4} + \frac{\theta}{2}\right) \quad (5)$$

For $k > c$,

$$S_k = \frac{k^2R^2\theta}{2} - \frac{(k-1)^2R^2\theta}{2} - 2A_1 - 2A_2.$$

$$= \frac{(2k-1)R^2\theta}{2} - 2A_1 - 2A_2 \quad (6)$$

It can be shown that $\theta = \frac{\pi}{2} - 2\cos^{-1}\left(\frac{W}{(k-1)R}\right)$ and

$$\alpha = \cos^{-1}\left(\frac{W}{kR}\right) - \cos^{-1}\left(\frac{W}{(k-1)R}\right) \quad (7)$$

Lemma follows from (3), (4), (5), (6) and (7).■

Nodes in the first strip do the maximum work by forwarding the packets from all the other nodes to the base station. Hence they die first. Therefore the network lifetime is bounded by the lifetime of the nodes in the first strip. Let $G_{from} = \sum_{i=1}^{K} C_i . i$ and $G_{to} = \sum_{i=1}^{K} C_i (\pi R^2 \delta - 1)$. $A_{SRPlife}$ denotes the network lifetime in number of iterations when SRP is run after T iterations. A_{life} denotes original lifetime without SRP. Aggregators in strip 1 receive $C_1(\Pi R^2 \delta - 1)$ packets from other nodes in strip 1 and $\sum_{i=2}^{K} C_i$ packets from nodes in other strips. They send $\sum_{i=1}^{K} C_i$ packets to the base station. The total number of sends performed by nodes in strip 1 is $T_1 = C_1(\Pi R^2 \delta - 1) + \sum_{i=1}^{K} C_i$. Whereas the total number of receives is $R_1 = C_1(\Pi R^2 \delta - 1) + \sum_{i=2}^{K} C_i$. Therefore the approximate lifetime of the network in iterations= The approximate lifetime in iterations of the closest nodes, $A_{life} = \frac{N_1 . PWR}{T_1 E_s + R_1 E_r}$ where N_1 is the number of nodes in strip 1.

Theorem 3.4: $A_{SRPlife} = \dfrac{N_1.PWR}{T_1E_s + R_1E_r + \left(\dfrac{C_1E_s + C_1(\pi R^2\delta - 1)E_r}{T}\right)}$ when SRP is performed every T iterations.

Proof: When SRP is used, clusterheads in strip 1 transmit C_1 packets to their neighbors (R_{sid} values) whereas nodes in strip 1 receive $C_1(\Pi R^2\delta$ -1) packets. The amount of energy consumed by nodes in strip 1 during SRP is $C_1E_s + C_1(\pi R^2\delta - 1)E_r$. Therefore an overhead (or the energy consumed by SRP) of SRP per iteration is $\dfrac{C_1E_s + C_1(\pi R^2\delta - 1)E_r}{T}$. Theorem follows.■

3.4 Results

To calculate the probability of success of the SRP method, let us assume that packet size is b bytes and the preamble of the packet is b/10 bytes in size. We say packets overlap whenever a small fraction of the packet overlaps. p_1 is the probability with which the adversary succeeds in guessing the time at which the original node may send the packet. Since WSN traffic is low and intermittent, it is safe to assume that there is quite a bit of idle time during which nodes do not transmit even during their allocated time slots. Hence it is difficult for an adversary to send a packet at approximately the time as the original node such that at least some part of packets overlap at the receiver. Therefore we consider values of p_1 from 0 to 0.3 for analysis.

When a collision happens, we assume that it cannot be detected even if only a fraction of the preambles overlap. p_2=1- P(collision cannot be detected) =1- (2*size of the preamble)/(2b)=1 - 0.1=0.9.

It is very hard for an adversary to guess that (i) the previous packet collided, (ii) a collision could not be detected at receiver and (iii) both packets were lost. But still we

give the benefit of doubt to the adversary and assume that the adversary always succeeds in guessing it i.e. q=1.

For the purpose of analysis we take p_2=0.9 [32]. It can be seen from figs. 3.11 and 3.12 that the probability of success is almost 1 when p_1 varies from 0.1 to 0.3, p_3 varies from 0.1 to 1 and m is 1 or 2. The probability of success increases as m increase or as p_3 increases. It also increases as p_1 decreases.

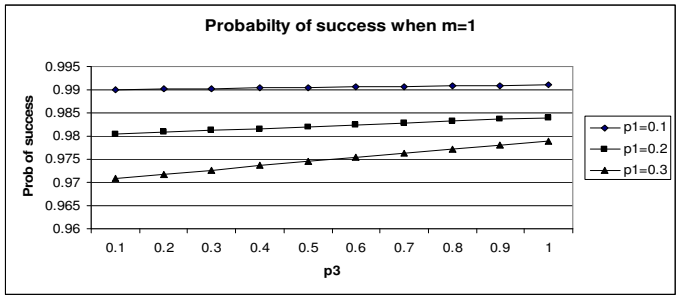


Figure 3.11: Probability of success when m=1.

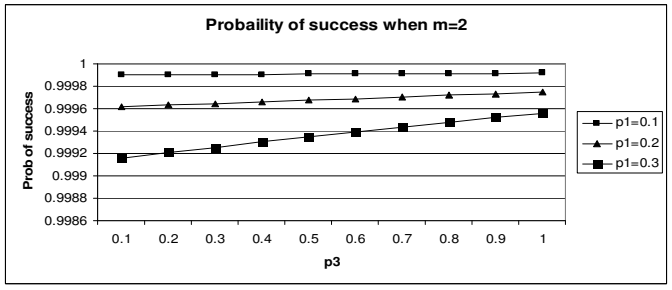


Figure 3.12: Probability of success when m=2.

We use the following parameters [33] to evaluate the overhead of SRP method.

Packet size=100 bytes
Max packets transmitted by radio/sec=48
Time for radio to transmit a packet=0.02083 sec
E_s =138.3112 J
E_r =54.9912 J

RC5 encryption/decryption energy per packet= 0.076 mJ.
Initial node energy=10^3 J

For N=15676, r=25, W=1000 we calculate an overhead of the SRP technique. In one iteration data packets from all the nodes reach the base station. The graphs below displays % decrease in lifetime when SRP is run after different number of iterations.

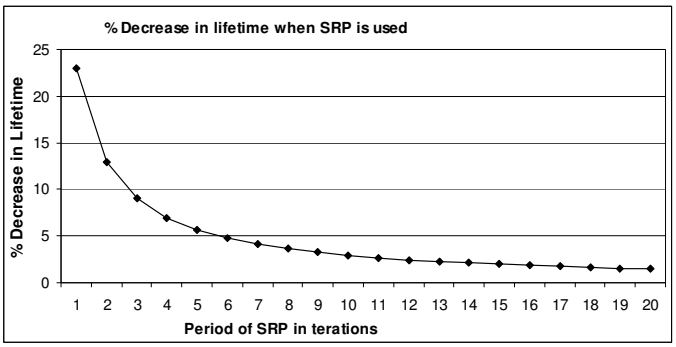


Figure 3.13: % decrease in lifetime against number of iterations after which SRP is run.

Network lifetime decreases by 23% when SRP technique is run after every iteration. The overhead is quite small and reduces the network lifetime by only 1% when SRP is run every 30 iterations.

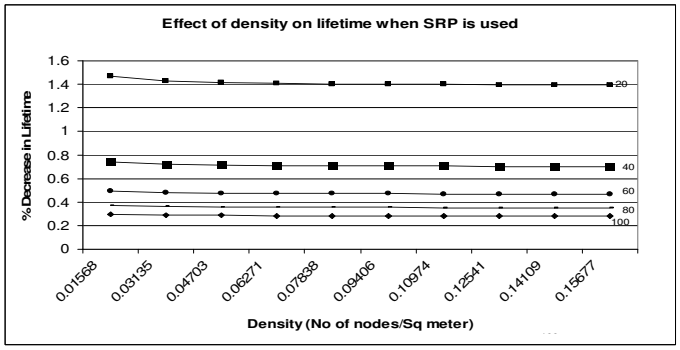


Figure 3.14: % decrease in lifetime for SRP against density of nodes when SRP is performed every 20, 40, 60, 80 and 100 iterations.

Figure 3.14 shows the percentage decrease in network lifetime against density when SRP is used after 20, 40, 60, 80 and 100 iterations. It can be seen that

increasing density does not increase the network lifetime. This is because all the nodes are busy during the iteration and they are placed uniformly. Of course if some data saving scheme, such as sleep – wake up schedules, is used then the overall network lifetime will increase by increasing density. But the effect of overhead of SRP on network lifetime will remain the same.

The overhead of the MG technique is negligible since it uses passive listening. Data packets are not transmitted. When both techniques can be used at the same time they can cover more scenarios in which attacks can occur.

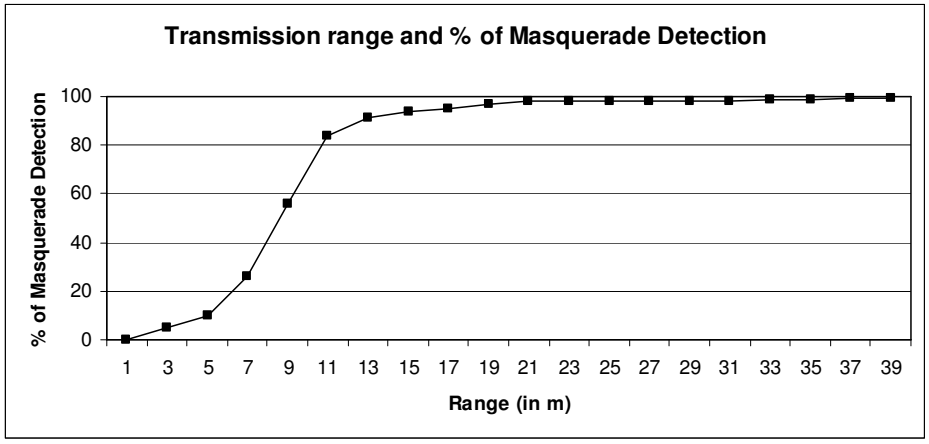


Figure 3.15: Effect of transmission range on percentage of detection of masquerade.

Figure 3.15 shows the effect of the transmission range of nodes on the probability of detection of masquerade for nodes randomly placed. We placed 100 nodes randomly in 100mX100m area. We assume that the adversary has the same transmission range as all the nodes. Every node is equally likely to be masqueraded. For each node, adversary finds out its neighbors and tries to masquerade as its neighbor. For each node, we place adversary at 100 random locations from where it can transmit a message to that node. It tried to attack all the nodes. Figure 3.15 shows

the percentage of detecting masquerade attack across the network against the transmission range of nodes. It is obvious that more peers guard each node if transmission range is greater. Therefore the percentage of detection increases with increase in transmission range. But a larger transmission range consumes more energy for transmission. Therefore higher percentage of detection can be achieved at the cost of more consumption of energy due to larger transmission range. The above technique also involves the cost of overhearing.

We use ns2 [46] to simulate WSN of 100 nodes in a square area of width 100m. Nodes have a transmission range of 10m. If masquerade attack is detected by MG method it costs 1.93 J of energy whereas an iteration of SRP costs 2.48 J of energy.

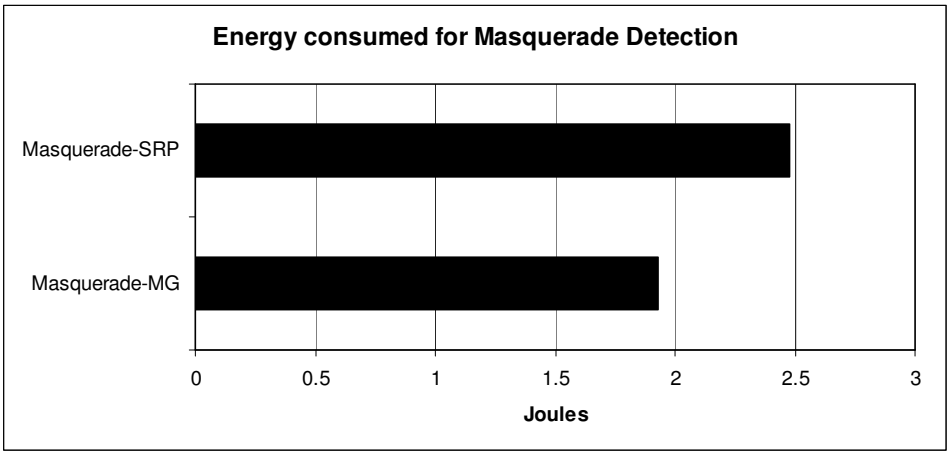


Figure 3.16: Energy consumed for masquerade detection.

3.5 Other Possible Approaches

These mechanisms can be implemented at multiple layers of a network stack to detect masquerade attack.

3.5.1 Using RSSI Value at Physical Layer

The problem of protecting radio interface (like prevention of eavesdropping and jamming) has been intensively researched for virtually all wireless networks and

many solutions have been proposed and deployed, such as spread spectrum communication and frequency hopping [34]. When a node receives a packet, it is difficult to find out if the packet came from the claimed sender unless explicit authentication is used. We try to address this problem by using Received Signal Strength Indicator (RSSI). Recently proposed embedded operating systems like TinyOS [35] provide functionality to get the RSSI value. For wireless medium, received signal strength is related to the distance between nodes.

We associate a neighbor with an estimated RSSI value. After deployment when nodes perform neighbor discovery, they record RSSI value for each neighbor. These recorded values can be used to detect intrusion afterwards. The packet received with RSSI value that is not in the range can be flagged. Similarly a sender can also be flagged for all further communication. Once intrusion is detected, various kinds of actions (like dropping a packet, flagging a neighbor etc.) can be taken. However in this chapter we focus only on *intrusion detection* and hence do not discuss solutions to handle intrusions. There are many factors like background noise, weather conditions etc. that can lead this approach to produce higher percentage of false positives. Therefore this approach should be used in combination with others (such as the ones proposed later in this chapter).

3.5.2 Techniques at MAC Layer

When scheduling based protocols are used for media access then there is a specific time slot allocated to each node. If an adversary wants to masquerade as some node it has to do that in the time slot allocated to that node. If adversary does not follow this schedule and tries to masquerade as some node at a time when that

node is not supposed to transmit, then nodes can detect an intrusion if they keep track of transmission schedule of other nodes. Below we show how this idea works for TDMA and S-MAC.

3.5.2.1 Time Division Multiple Access (TDMA)

TDMA is a digital transmission technology that allows a number of users to access a single radio-frequency channel without interference by allocating different time slots to different users within each channel.

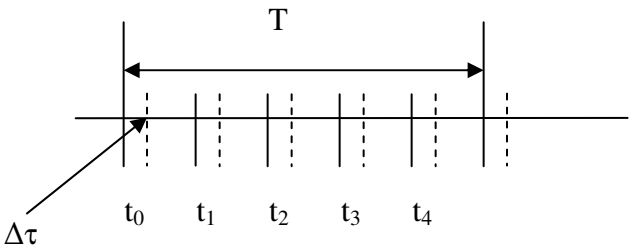


Figure 3.17(a): TDMA schedule for nodes - clock drift of $\Delta\tau$.

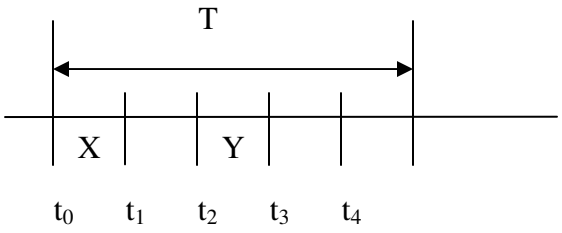


Figure 3.17(b): TDMA schedule for nodes - no clock drifts.

Suppose nodes keep track of TDMA schedules of other nodes that they communicate with. Node X is allocated time slot t_0 and node Y is allocated time slot t_2 (see figure 3.17(a)). For simplicity we assume that all time slots are of length τ. Suppose an adversary tries to send a packet with sender field set to X in time slot t_2. For a node that receives a packet with source field in the packet set to X in a time slot

that is not allocated to X, is an anomaly. In this way the data used by TDMA protocol can be used to detect intrusion. If an adversary sends a packet in the time slot t_0 by changing the source field to X then that packet will not be detected. If we assume that the adversary sends packet randomly in any time slot of length τ (τ =T/n) and there are n nodes sharing the medium (in n slots), then the probability of detection is (n-1)/n. The probability of false negatives is 1/n. If clocks are synchronized, there will be no false positives. If there is a clock drift of $\Delta\tau$, then the probability of false positives is $(\Delta\tau*n)/n*\tau = \Delta\tau/\tau$. The probability of false negatives is same as the probability of false positives.

3.5.2.2 S-MAC

Many MAC protocols like S-MAC have been proposed which use sleep/wake-up schedule for energy conservation (see figure 3.18). If those protocols are in use and node A receives a packet with source field set to X at a time when node X should be sleeping, then node A can easily detect that the packet is sent by an adversary. Node A can detect this intrusion because it is an anomaly. Above we propose anomaly detection technique for schedule-based MAC protocols. These techniques use the available data and hence incur very little overhead, which suits resource constrained nature of WSNs.

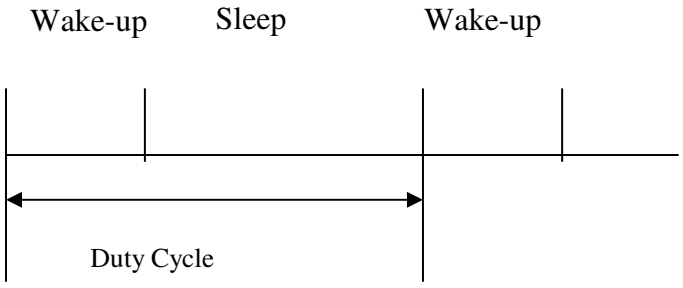


Figure 3.18: S-MAC sleep/wake-up schedule for a node.

3.5.3 Techniques at Application Layer: Use of Round Trip Time

At application layer, round trip time can be used for intrusion detection for a bi-directional communication. We associate round trip time with a neighbor. If the round trip time for some neighbor is not in an estimated range, that neighbor can be flagged. Similar to RSSI technique, there are many factors like background noise, weather conditions etc. that can lead this approach to produce large number of false positives. Hence this approach should be used in combination with others.

3.6 Summary

We proposed two lightweight techniques for detecting masquerade attack. Results of this chapter appear in [40, 57]. Our solutions take into account important WSN characteristics such as coverage, connectivity, aggregation and communication patterns. Our main results can be summarized as follows.

1. Both methods are independent and compliment each other in preventing attacks.
2. MG method fails to protect nodes that are one hop away from the border or when attacker has shorter communication range than sensor nodes. SRP overcomes these drawbacks at a reasonable cost.
3. MG method incurs insignificant overhead as it uses only passive listening. SRP decreases network lifetime by only 1% when it is run after 30 iterations. Overhead of SRP is minimal.
4. The probability of success is very high for SRP.
5. We also propose use of RSSI values, MAC schedules and round trip time for anomaly detection which leads to the detection of masquerade attack.

CHAPTER 4

DETECTION OF SYBIL

Abstract

In this chapter we extend two solutions, the MG method and the SRP method, discussed in chapter 3 to detect the Sybil attack. Proposed methods are independent and complimentary.

4.1 Introduction

For a Sybil attack we assume that the attacker poses *id*s from A_1 to A_d such that $[A_1, A_d] \in \{1, 2, 3, \ldots, N\}$. If attacker poses *id* other than 1 to N then it can be detected easily [40] because new nodes are added securely and nodes know their neighbors. Therefore attacker must pose *id* that is from 1 to N. We consider two cases for Sybil attack. (i) An attacker disables nodes A_1 to A_d and (ii) An attacker does not disable nodes A_1 to A_d.

We extend the two solutions discussed in chapter 3 to detect Sybil attack - (i) MG method and (ii) SRP method. We discuss MG method next.

4.2 Mutual Guarding (MG) for Sybil Detection

When nodes are static and new nodes are added securely (as explained later), the presence of the foreign entity can be detected easily. If any node detects the presence of a node that is not its neighbor, then it suspects that an attacker is present which may be Sybil or may be posing a masquerade attack (attacker poses only one false *id,* hence masquerade is a special case of Sybil). Therefore attacker must assume *id*s of only neighbors. Attacker may or may not disable the actual nodes it assumes *id*s of.

1. Let us assume that an attacker poses the *id*s A_i for $i \leq d$ (for a constant d) but does not disable nodes A_i. There are 2 possibilities:
   (i) Actual node A_i is in the neighborhood of an attacker and can directly receive packets from the attacker. In this case, actual node A_i detects the presence of the Sybil node.
   (ii) Actual node A_i is not in the neighborhood of an attacker and cannot directly receive packets from the attacker. There is at least one neighbor of the attacker that has not directly received any packet from A_i before. Thus it suspects Sybil attack upon receipt of a packet from A_i.
2. If attacker disables the actual nodes then it becomes difficult to detect the presence of the attacker. But still it can be detected in some cases as shown next. Let us assume that node A_1 was compromised physically, its code was altered and it poses identities of nodes from A_1 to A_d. Nodes A_2 to A_d are disabled by the adversary. If any neighbor of nodes A_2 to A_d notice that those nodes are disabled and at the same time some other node can listen to communication from some node with *id* such that $A_2 \leq id \leq A_d$, then base station suspects Sybil. Base station sends a query to find if a node is active after receiving messages that some nodes have been disabled.

Consider sensor nodes deployed (as shown in figure 3.6) in a square area of length W. This deployment uses the minimum number of nodes to cover the whole area. The deployment shown here is similar to the one presented in [2]. We consider this deployment only for analyzing the proposed Sybil detection technique. Let the number of nodes in each row be N_r. Then $N = N_r^2$ and $W=(N_r + 1)r$. Therefore N_r

$= \left\lceil \frac{W}{r} \right\rceil - 1$. Nodes closest to the border are not completely covered. Hence they will not always be able to detect an attacker posing as their neighbor.

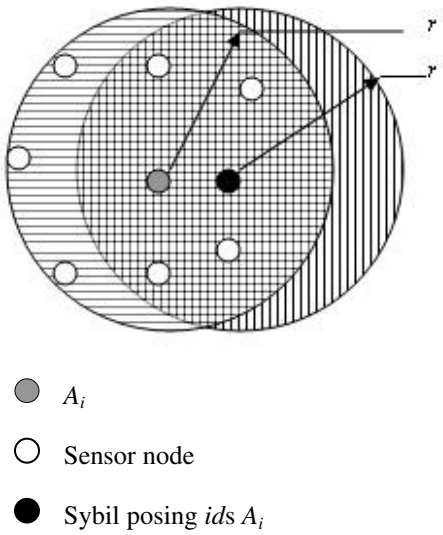


Fig 4.1: Actual node overhears its *id* being assumed by Sybil.

Corollary 4.1: If an attacker has a transmission range at least as long as the deployed sensor nodes and it does not disable any node, then for the optimal deployment as shown in figure 3.6, the attacker can assume the *id*s of approximately $8N_r - 16$ nodes if it finds proper place to transmit from and it can perform Sybil attack on approximately $4N_r - 4$ nodes.

Proof: Theorem 3.1 establishes this result for masquerade attack. It can be extended for Sybil attack.■

It can be observed that $O(\sqrt{N})$ nodes are vulnerable to Sybil attacks even when the MG method is used. Below we discuss a distributed mechanism that guarantees that our solution does not generate a false alarm. The base station initiates the distributed

mechanism. We assume that the attacker has only as powerful resources as the sensor nodes.

Let the node *A* (and it's another suspected identity *A'*) be in the $k^{'th}$ strip (please see figure 3.10(a)). Base station finds two nodes - let us say *X* in $(k\text{-}1)^{'th}$ strip and *Y* in $(k\text{+}1)^{'th}$ strip. *X* and *Y* are such that they can listen to messages sent by *A* or *A'*only but not both. Nodes *X* and *Y* can be found as below (see figure 4.2). The node that reports the presence of Sybil starts a distributed mechanism to find nodes *X* and *Y*. It simply broadcasts a control packet in the neighborhood to find out such nodes. Nodes that receive packets only from *A* or *A'* reply with another control packet. It may not always be possible to find such nodes especially if *A* and *A'* are very close to each other.

Once nodes *X* and *Y* are found, nodes *A* and *A'* are asked to transmit packets continuously for a finite interval of time say *t*. Note that this will not cause collisions because of exposed node problem [43]. Let *b* be the capacity of the antenna in kbps, *p* be the packet size in bits. Each of the nodes *X* and *Y* should have received at least $1000bt/p$ packets if the suspected node is not Sybil. If the suspected node is Sybil then it cannot transmit more than $500bt/p$ packets to both nodes *X* and *Y*. *X* and *Y* send the number of packets received from two identities to the base station which detects whether the node is Sybil.

MG method does not guard nodes that are located near the border from the Sybil attack. For randomly placed nodes, even the nodes that are inside may not be completely covered and are not guarded from the Sybil attack. Our next proposed solution SRP does not have these drawbacks.

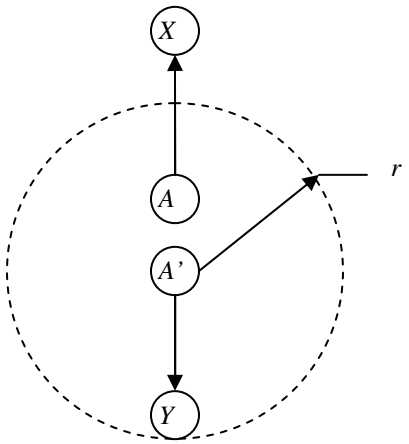


Figure 4.2: Suspected *id*s *A* and *A'* tested for Sybil.

4.3 SRP: Verification of the Number of Packets Sent and Received for Sybil Detection

As stated earlier, if the attacker assumes the *id* of a node that is not a neighbor, then it is an anomaly and the attacker can be detected easily. Our proposed technique can now be described in detail as follows:

Let d be a node. Let s_i denote its i^{th} neighbor for $i > 0$. The following test is performed every T iterations. s_i keeps track of the distinct number of packets sent (S_{sid}) to d during the time period that lasts for T iterations. Then d broadcasts a single packet containing the number of packets it received from its neighbors (R_{s1d}, R_{s2d}, R_{s3d}, …). If $R_{sid} > S_{sid}$ then s_i concludes that there is a Sybil node that assumes the *id* s_i. Note that we assume for simplicity of discussion that the link layer is reliable which implies that packet losses due to noise, collisions etc. are handled reliably and a packet sent is received (albeit maybe after retransmits).

Attacker can perform a DoS attack on the above solution but it can be detected easily. If adversary broadcasts the same packet that d broadcasts then receivers will receive two such packets (one from adversary and one from d) in a time period that

lasts for T iterations. This is an anomaly and it can be guessed that adversary is performing DoS attack on SRP.

4.4 Results

We use ns2 [46] to simulate WSN. Simulation parameters are shown in figure 4.3. We placed 100 nodes randomly in a square area of length W=100m. Nodes have transmission range of r=10m. Nodes that are in communication range form clusters.

W=100
r=10
N=100
Packet size=100 bytes
Energy to send a packet, E_s =1.38 J
Energy to receive a packet, E_r =0.549 J
RC5 encryption/decryption energy per packet= 0.076 mJ.
SHA1 One-way function per packet=0.065 mJ.
PWR, Initial node energy=1000 J

Fig 4.3: Simulation parameters.

For verification of Sybil, we give two suspected nodes 0.5 second time to transmit packets to two nodes. Each of the nodes that perform a test spends maximum of 22 J of energy. If Sybil attack is detected by MG method it costs 1.93 J of energy whereas an iteration of SRP costs 2.48 J of energy.

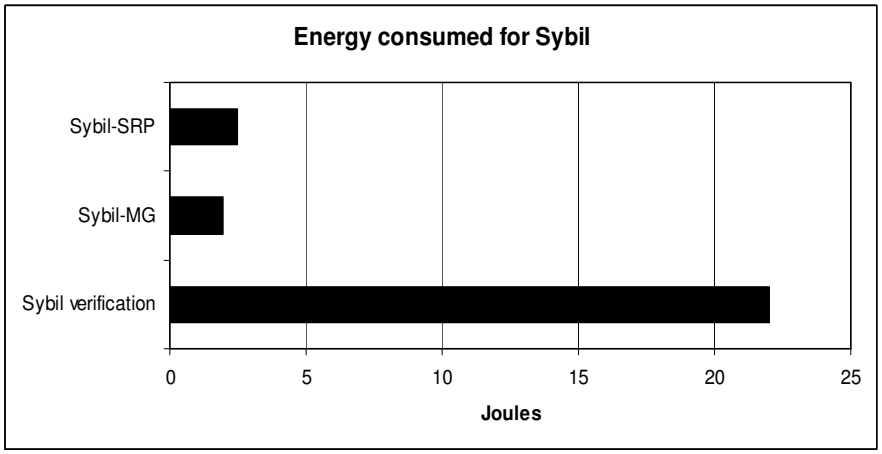


Figure 4.4: Energy consumed for detecting Sybil attack.

4.5 Summary

In this chapter we propose two independent and complimentary solutions to detect Sybil attack. MG method costs 1.93 J of energy whereas an iteration of SRP costs 2.48 J of energy. Results of this chapter appear in [56].

CHAPTER 5

DETECTION OF PACKET-DROPPING ATTACKS

Abstract

Denial-of-service (DoS) attacks on wireless sensor networks (WSN) can deplete network resources and energy without much effort on the part of an adversary. Packet-dropping attacks are one category of DoS attacks. Current techniques for detecting such attacks in ad hoc networks need to monitor every node in the network. Once they detect malicious nodes that drop packets, a new path has to be found that does not include them. In this chapter, we propose a solution, to detect packet dropping, called DPDSN. It identifies paths that drop packets by using alternate paths which WSN finds earlier during route discovery. Responding to a packet-dropping attack incurs no additional cost because one of the alternate paths is utilized for all subsequent communication. DPDSN does not require monitoring individual nodes, making it feasible for WSN. We formulate the probability of success and failure of DPDSN in the presence of malicious nodes that drop packets. We compare our approach with existing techniques. Our analysis found that the overhead of DPDSN is at most $O(\sqrt{N})$ *for a two-dimensional grid network of N nodes. We show that the overhead of DPDSN for a WSN with 100 nodes is less than 3% of energy consumed on route discovery when using DSR or Directed Diffusion routing protocols.*

5.1 Introduction

In this chapter, we address the problem of detecting packet-dropping attacks in WSN. Apart from malicious intent; there can be other reasons of packet dropping like

collisions, buffer overflows, congestion, etc. It is important to find solutions that take these factors into account, for example, to prevent false alarms.

Existing solutions [10, 12] for detecting packet dropping in ad hoc networks work by monitoring individual nodes. Sleep-wakeup schedules followed by nodes in a WSN [24] make continuous monitoring impractical. Also, monitoring individual nodes is too expensive for WSN.

Our approach, called DPDSN (Detection of Packet-Dropping attacks for wireless Sensor Networks), uses the observation that alternate routing paths are readily available in WSNs, which are typically dense. DPDSN monitors paths and detects whether any node on a path drops packets. Once we detect such an event, we switch to an alternate path for communication. We always keep an alternate path ready to minimize the switching delay. The cost of finding an alternate path is minimized by having it embedded in the route discovery of source-initiated and receiver-initiated routing protocols such as the ones proposed in [36, 23].

Keeping alternate paths readily available is justified even if packet-dropping attacks are not detected. First, the alternate paths can be used for load-balancing transmissions. Second, uneven consumption of energy is a biggest threat to lifetime of a WSN because it can partition the network. Use of alternate paths for transmission can protect nodes on the original path from expending all their energy too soon.

DPDSN can be extended to detect individual nodes that drop packets. We do so only if there is a real need, because finding such nodes is costly for resource-constrained WSNs.

5.2 The DPDSN Approach

This section describes the DPDSN approach, including detection of compromised paths, embedding alternate path discovery in route discovery, overhead analysis, and discussion of malicious node discovery.

5.2.1 Detection of Compromised Paths

Our detection mechanism uses an alternate path between a source and a destination. We propose that the process of finding an alternate path be embedded in the route discovery phase of routing protocols like DSR [36] and Directed Diffusion [23]. Ideally, the alternate path does not have any node in common with the original path. We assume that the source and the destination are not malicious or compromised.

Packet loss can be caused by congestions due to heavy traffic, collisions at link layer, buffer overflows, etc. In WSNs, congestions and buffer overflows seem unlikely to happen because of low traffic rates. Reliable MAC protocol rules out collisions as a reason for packet dropping. Assuming reliable MAC protocol and assuming low traffic rates, the main possible reason for packet losses in WSN is malicious non-forwarding or packet dropping by an adversary or compromised nodes. We focus on this to detect paths that drop packets.

Compromised paths can be detected as follows. Let n_s be the number of packets sent by the source in a given period of time. Using the alternate path found during route discovery, the source periodically requests the destination to send the number of packets received, n_r. In DPDSN, the source sends query to the destination using an

alternate path, requesting it to send n_r. Algorithm in figure 5.1 outlines the detection approach for the currently used path.

Detect_compromised_path(*s*, *d*)

begin

Get n_s, n_r.

while (TRUE)

if $n_s - n_r > 0$ **then**

Guess that packets are being dropped by malicious nodes on the source-to-destination path.

return TRUE

else

return FALSE

Wait till next verification cycle.

end

Figure 5.1: Detection of a packet-dropping path.

The alternate path is used not only to verify whether n_s=n_r. If we find that packets are dropped by the original path, it can also be used for all subsequent communication.

5.2.2 Embedding Alternate Path Discovery in Route Discovery

Finding an alternate path during route discovery is a challenging problem and finding an optimal alternate path is beyond the scope of this chapter. We address the embedding problem heuristically and show possible approaches for DSR and Directed Diffusion.

5.2.2.1 Embedding in DSR

Let us consider an example shown in figure 5.2(a) and figure 5.2(b) for DSR. Node 1 is the source whereas Node 8 is the destination. Node 1 sends out a route request packet, which floods the network as shown by broken-line arrows in figure 5.2(a). Node 8 responds by sending route reply packet as shown by single-line arrows in figure 5.2(b). Double-line arrows in figure 5.2(b) indicate an alternate path that can be used later to verify the number of actual packets received by the destination. In this case, Node 8 determines if there is an alternate path to Node 1. The destination node finds an alternate path that does not have any node in common with the nodes that are on the original path. Next, we need to discuss the difficulties one may face while addressing this problem. In some cases it may not be possible to find an alternate path. For example, in figure 5.2(c), the original path from Node 8 to Node 1 is through Nodes 5 and 2. The alternate path from Node 8 to Node 1 includes Nodes 4, 5 and 2. In this case, detecting whether packets are dropped on a path from Node 8 to Node 1 will not work if Node 2 drops packets. Note that even if we detect that Node 2 drops packets, there is no alternative path to use to avoid packet dropping on a path from Node 1 to Node 8.

There can be cases for which even if there is an alternate path from the destination to the source, the destination cannot find it from the route request packets it receives. It happens because some information is lost when intermediate nodes forward a route request packet after receiving multiple route request packets from different neighbors. In figure 5.2(a), Node 7 receives route request packets from Nodes 4 and 6 but it

forwards only one of the packets to Node 8. In general suppose *s* is the source, *d* the destination and *j* the only neighbor of d. Also, assume that *j* receives multiple route request packets from its predecessors. Node *j* will forward only one route request packet to *d*. In this case there is no alternate path from *d* to *s* because *j* is the only neighbor of *d* that can forward packets from *s* to *d* or *d* to *s*. But there may be an alternate path from *j* to *s*. After receiving only one route request packet (of course from *j*), node *d* may ask node *j* to find an alternate path from *j* to *s*, if one exists. If *j* receives only one route request packet, then it can ask its predecessor k to find an alternate path from *k* to *s*.

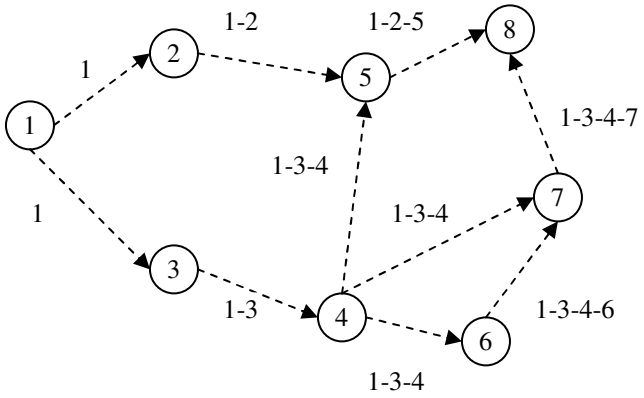


Figure 5.2(a): Source (Node 1) floods the network with DSR route request packet. Node 8 is the destination.

There can be a case in which node d receives multiple route request packets but it cannot find *node-disjoint* alternate path to reach *s*. It may find an alternate path to reach some intermediate node *m*.

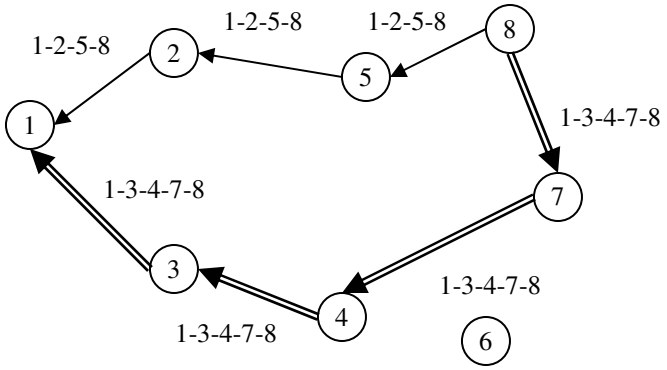


Figure 5.2(b): Destination (Node 8) sends route reply packet (single-line arrows). Destination finds alternate path (double-line arrows).

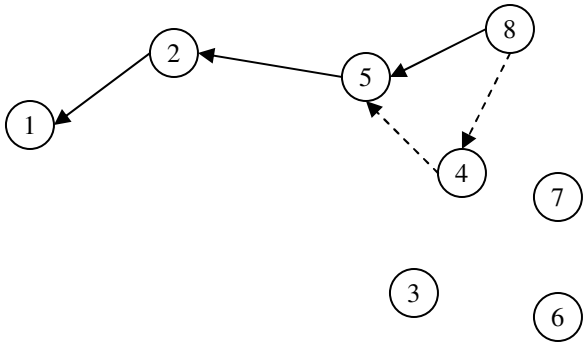


Figure 5.2(c): Alternate path from Node 8 to Node 1 does not exist.

Algorithm in figure 5.2(d) outlines a heuristic process of finding an alternate path when DSR is used for route discovery. A straightforward solution is to perform route discovery using DSR and mark the edges of the original path. It incurs significant cost due to flooding. A better heuristic approach would be to keep two route requests at every node when a node receives multiple route requests. One of the route requests is used for establishing the path and second one will be used for alternate path. The algorithm is outlined in figure 5.2(d). Obviously this algorithm is just one approach and my not result in an efficient alternate path. Finding optimal alternate path is a

challenging problem as mentioned above. The described algorithm suffices for this chapter to address packet-dropping attacks for WSNs.

Find_alternate_path(*s*, *d*, AP)

begin

// Let N be the number of nodes. Each node stores 2

//route requests. Node j stores its route replies in

//R[j, 0] and R[j, 1]. predecessor[x] is a function that

//defines the set of nodes that can send route request

//to node x. AP is initialized to {d}.

if (d=s) **then** return.

if (|predecessor [*d*]|=1) then //No alternate path to s exists.

j= predecessor [*d*]

else

if($j \in$ R[*d*, 1])

j= predecessor [*d*] //note *j* not in R[*d*, 0]

AP=AP $\cup$ {*i*}

Find_alternate_path(*s*, *d*, AP).

end

Figure 5.2(d): A heuristic algorithm to determine alternate path.

5.2.2.2 Embedding in Directed Diffusion

Let us consider the alternate path discovery problem for Directed Diffusion, a receiver-initiated protocol. In Directed Diffusion, the destination/sink (Node 8) floods the network in search of some data (called an *interest*) as shown by broken-line

arrows in figure 5.3(a). Whenever that message reaches the source (Node 1), it floods the network back as shown by solid line arrows. Then, the source reinforces only one path from the source to the sink, which is shown by single-line arrows in figure 5.3(b). This path is used for all future communication. We modify the reinforcement step similar to the ideas used for DSR to identify an alternate path. The double line arrows in figure 5.3(b) show an alternate path discovered during reinforcement step.

5.2.3 Analysis of Overhead for Finding Alternate Paths

We analyze the overhead for finding alternate paths for DSR and Directed Diffusion. Let the cost of sending a packet be E_s and the cost of receiving a packet be E_r. *Let* N be the number of nodes, m be the average number of node neighbors (*neighbor* of a node is any other node within its broadcast range). Let p be the length of the path.

5.2.3.1 Analysis for DSR

For DSR, the approximate cost of route request and route reply is $N(E_s + mE_r)$ and $p\ (E_s + E_r)$, respectively. The cost of finding an alternate path is $p(E_s + E_r)$. Therefore, the ratio of the cost of finding an alternate path to the cost of DSR path discovery is $p(E_s + E_r) / (N\ (E_s + mE_r) + p(E_s + E_r))$.

The overhead ratio is plotted against p and E_r/E_s in figure 5.4. E_r/E_s is the ratio of the energy consumed for receiving a packet to the energy consumed for sending a packet. We assume N=100 and m=6. The reason for this value of m is as follows. The maximum coverage and the maximum number of neighbors for each sensor are provided by beehive configuration, in which there are 6 neighbors per node (except nodes on the boundary) [37].

Sending a packet consumes much more energy than receiving it. For MICA2 mote sensors [38], sending a packet consumes 81 mW energy, whereas receiving it consumes 30 mW energy [39], that is E_r/E_s is 0.37. To cover the E_r/E_s range well, we chose values of E_r/E_s to be from 0.25 to 0.5 as shown in figure 5.4. Overhead increases as path length or E_r/E_s increases. Still it is no higher than 6% for a path of length 13.

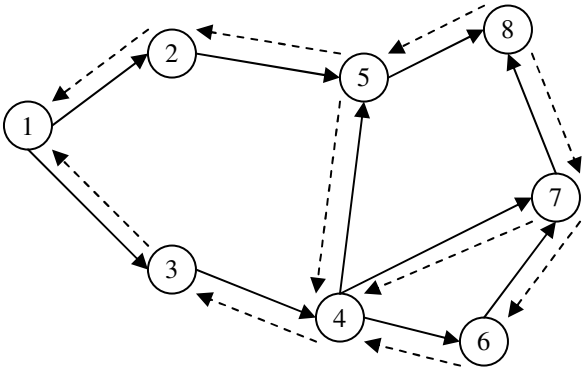


Figure 5.3(a): Source (Node 1) sends out interest (solid lines). Sink (Node 8) replies back (broken lines).

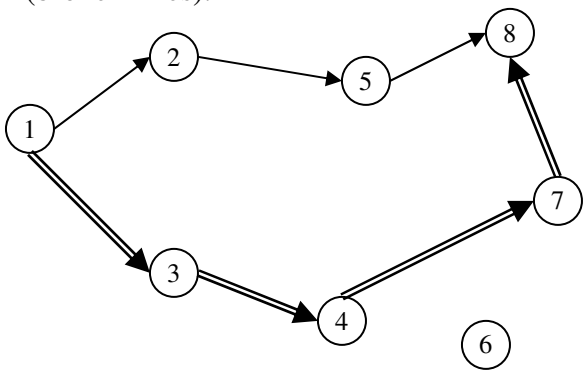


Figure 5.3(b): Source (Node 1) reinforces a path (single-line arrows) to reach sink (Node 8) and finds alternate path (double-line arrows).

5.2.3.2 Analysis for Directed Diffusion

For Directed Diffusion, the approximate cost of path discovery is $2N(E_s + mE_r) + p(E_s + E_r)$. The cost of finding an additional path is $p(E_s + E_r)$. Therefore, the ratio of

overhead of finding an alternate path to the cost of Directed Diffusion path discovery is $p(E_s + E_r)/ (2N(E_s + mE_r) + p(E_s + E_r))$.

The overhead ratio is plotted against p and E_r/E_s in figure 5.5. We assume N=100 and m=6. For reasons explained above, we again chose values of E_r/E_s from 0.25 to 0.5 as shown in figure 5.5. Overhead increases as path length or E_r/E_s increases. Still it is no higher than 3% for a path of length 13.

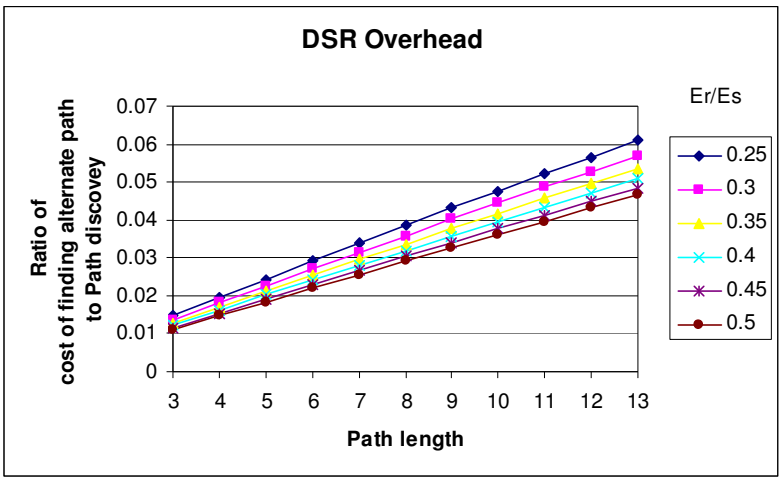


Figure 5.4: Ratio of overhead of DPDSN to DSR path discovery as a function of path length and E_r/E_s.

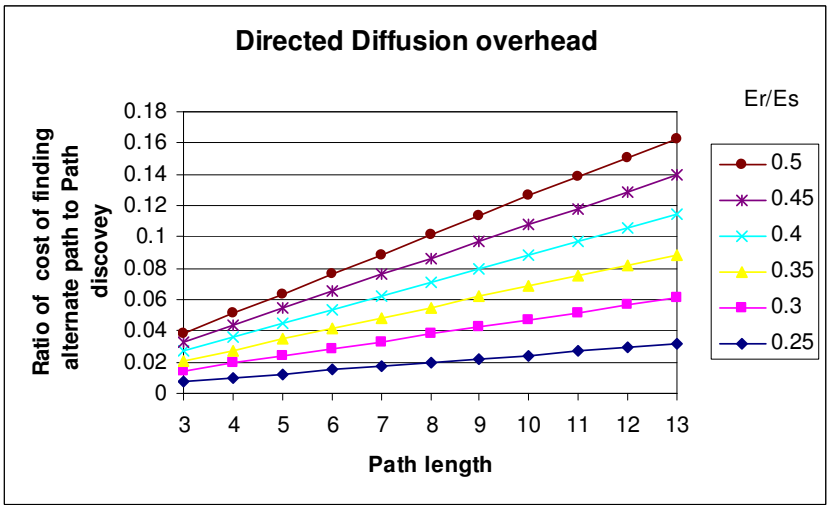


Figure 5.5: Ratio of overhead of DPDSN to Directed Diffusion route discovery as a function of path length and E_r/E_s.

5.2.4 Probability of Success for DPDSN

DPDSN works when we can correctly determine whether the original path is dropping packets. DPDSN *succeeds* whenever an alternate path does not have any malicious nodes that drop packets. In the following lemma we assume that N nodes are placed randomly, and M of them are malicious and drop packets.

Lemma 5.1: Assuming that we always find an alternate path, the probability of success for DPDSN is $\binom{N-M}{p} \div \binom{N}{p}$ where N is the number of nodes in the network, M is the number of malicious nodes that drop packets, and p is the length of the alternate path.

Proof: Let P(k) be the probability of finding k malicious nodes on a path of length p. Note that P(0) is the probability of success for DPDSN.

For $k \geq 0$, P(k) = (No. of ways of selecting k malicious nodes) * (No. of ways of selecting p-k non-malicious nodes) / (No. of ways of selecting p nodes from N nodes)

There are $\binom{N}{p}$ ways of selecting p nodes from N nodes, $\binom{M}{k}$ ways of selecting k malicious nodes from M malicious nodes and $\binom{N-M}{p-k}$ ways of selecting p-k non-malicious nodes from N-M nodes.

Hence, $\mathrm{P}(k) = \binom{M}{k}\binom{N-M}{p-k} \div \binom{N}{p}$ for $k \geq 0$

The probability P(0) of finding no malicious node on the alternate path is:

$\mathrm{P}(0) = \binom{N-M}{p} \div \binom{N}{p}$. ■

Obviously, the probability of failure of DPDSN is 1-P(0).

```
Detect_compromised_nodes(s, d, V)
    begin
    // Let s, s+1, s+2, … , d-1, d be the nodes that are
    // on the path from source s to destination d.
    // V is the set of nodes that drop packets.
        if d-s=2 and Detect_compromised_path(s, d)
        = TRUE then
            V=V ∪ {s+1}
        elseif Detect_compromised_path(s, d) = TRUE then
            if Detect_compromised_path (s, [s+(d-s)/2]) = TRUE then
                Detect_compromised_nodes(s, [s+(d-s)/2], V)
            if Detect_compromised_path ([s+(d-s)/2] + 1, d) = TRUE then
                Detect_compromised_nodes([s+(d-s)/2] + 1, d, V)
    end
```

Figure 5.6: Algorithm detecting compromised nodes.

5.2.5 Finding a Specific Node Dropping Packets

DPDSN can be used to detect specific nodes that drop packets. Assume, without a loss of generality, that the nodes on the path from source *s* to destination *d* are labeled *s*, *s*+1, *s*+2, .. ,*d*-1, *d*.

We can detect whether packets are being dropped on this path using the mechanism discussed in section 5.3.2. We describe that algorithm as the procedure

Detect_compromised_path (s, d) shown in figure 5.1. It returns TRUE if packet dropping by a malicious node is detected on the currently used path from s to d.

Let Detect_compromised_nodes(s, d, V) denote a procedure that performs a binary search for packet-dropping nodes on a path from s to d (figure 5.6). It stores detected packet-dropping nodes in set V. $[x]$ represents integer portion of a real number x (the *floor* function).

5.3 Comparison of DPDSN with Existing Approaches and Results

In this section, we estimate the overhead of other approaches and compare them with DPDSN. We calculate the overhead in terms of the number of messages sent and received. We compute the overhead for all the approaches for just the detection of packet dropping by a path.

The other existing approaches incur separate cost for the *response*. Response includes finding alternate path avoiding nodes dropping packets. It is important to note that for DPDSN the cost for response is *included* in the cost of bad path detection. The reason is that the *same* alternate path can be used later for transmitting packets.

5.3.1 Analytical Results on DPDSN Overhead

The destination is queried periodically by the source verifying the number of packets received by the destination. The following lemma considers overhead for one query period T, that is for a single verification cycle.

Lemma 5.2: For a single verification cycle, the overhead for DPDSN is $O(p)$ messages for a network of N nodes where p is the path length.

Proof: Overhead of DPDSN is the cost of establishing an alternate path, and sending and receiving a query (verifying the number of packets received by the destination) that uses this path. For any path, one message (from the destination to the source) has to be sent during the route reply phase and two messages (one from the source to the destination and another from the destination to the source) have to be sent for verifying the number of packets received. Every node on the path sends and receives these 3 messages. Therefore, for a path length of p hops, the cost is 3p send messages and 3p receive messages. This cost is for a single verification cycle. Hence, the overhead is $O(p)$. ■

Corollary: For a single verification cycle, the overhead for DPDSN for a two-dimensional grid sensor network of N nodes is $O(\sqrt{N})$ messages where N is the number of nodes.

Proof: Overhead of DPDSN, for a path length of p hops, is 3p send messages and 3p receive messages. This cost is for a single verification cycle. For a two-dimensional grid sensor network of N nodes, maximum value of p can be $2\sqrt{N}$. Therefore, the maximum cost of DPDSN is $6\sqrt{N}$ send messages and $6\sqrt{N}$ receive messages. Hence, the overhead is $O(\sqrt{N})$. ■

5.3.2 Analysis of DPDSN Overhead for Finding Specific Node Dropping Packets

For a path of length p, determining whether packets are dropped on the path costs $3p$ send messages and $3p$ receive messages. Detecting a specific node that drops packets will cost $6p$ send messages and $6p$ receive messages. In the worst case, all nodes on the path except the source and the destination can be malicious and detecting them all one after another will cost $3p.logp$ send messages and $3p.logp$

receive messages. Detecting individual nodes that drop packets incurs significant cost for WSNs, so Detect_compromised_nodes() should be used only if there is a real need of such fine detection.

5.3.3 Comparison with Other Approaches

The following analysis uses j as the number of packets sent, and δ as the number of packet-dropping instances detected in time period T.

For the same time interval and path of length p, Huang and Lee's approach [HL03] needs $p/2$ dedicated monitor nodes placed on a path alternately, such that each of the $p/2$ monitors nodes guards the remaining $p/2$ nodes. The monitors actively participate in the communication and they have to take into account each and every message sent in a period of time T for detection of packet dropping. If j messages are sent in a period of time T, the cost is jp send messages and jp receive messages.

If j messages are sent over a path of length p in a period of time T, Marti's approach incurs a cost of jp receive messages.

The approach taken by Buchegger *et al.* uses Marti's watchdog mechanism apart from the trust manager, the reputation system and the path manager. Reporting detected attack to other watchdogs will cost p send messages and p receive messages. If δ instances are detected in a period of time T, reporting local detections to other watchdogs will cost δp send messages and δp receive messages.

The technique proposed by Michiardi *et al.* also uses Marti's watchdog mechanism apart from the distributed cooperative reputation mechanism. Therefore, the cost is even higher. These kinds of approaches are too expensive to be suitable for resource-constrained WSNs.

Figure 5.7 summarizes the costs for different techniques in terms of the number of massages sent and received for detecting packet dropping over a path of length p in a time period T.

Techniques for detecting packet-dropping attacks	Overhead (in time period T)
DPDSN	$3p$ send and $3p$ receive messages
Huang and Lee [13]	jp send and jp receive messages
Marti *et al.* [10]	jp receive messages.
Buchegger *et al* . [20]	$(jp+ \delta p)$ receive and δp send messages
Michiardi *et al.* [12]	$(jp+ \delta p)$ receive and δp send messages + the cost for reputation mechanism

Figure 5.7: Comparison of techniques for detecting packet-dropping.

5.3.4 Experimental Results on DPDSN Overhead

In order to further validate our approach, we wrote a simulation program in C program to simulate the following communication scenario. The simulated WSN consisted of 100 nodes that were placed randomly in an area 100m by 100m. Base station is situated in one corner. The base station is the message source in all cases.

We set the radio range of nodes to 12m because this assures that each node has approximately 6 neighbors (based on random placement of nodes), which is beneficial for communication. For MICA2 mote sensors [38], sending a packet consumes 81 mW energy, whereas receiving a packet consumes 30 mW energy [39].

We varied the path length from 3 to 13 hops. We randomly selected malicious nodes. We assumed that only malicious nodes drop packets. The alternate path was used to verify the number of packets sent over the original path.

Based on the results, we calculated the ratio of overhead of DPDSN to the cost of discovering path using DSR and Directed Diffusion. Figure 5.8 and figure 5.9 show the overhead for DSR and Directed Diffusion. We found that the overhead for DPDSN is proportional to the path length. This observation is consistent with the analytical results shown in figure 4.7 and Lemma 5.2. This observation is also validated by figure 5.4 and figure 5.5, which plot the overhead for different values of E_r/E_s. The overhead is independent of the number of packets sent in a given period of time. Figure 5.10 shows the probability of success of DPDSN.

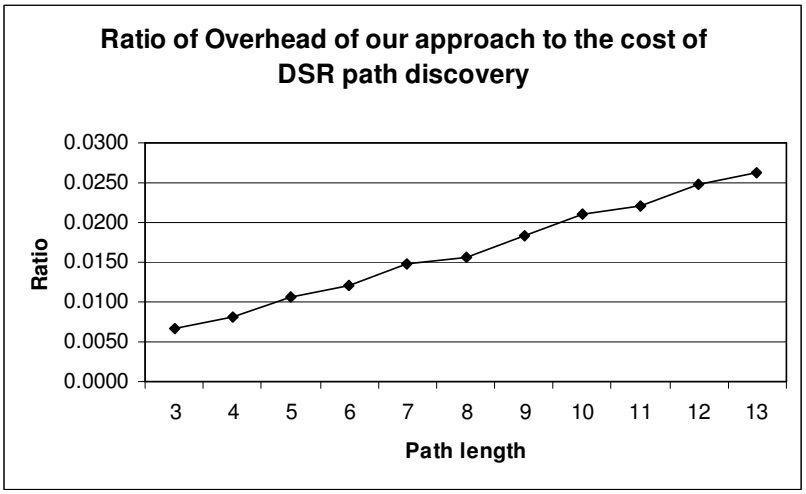


Figure 5.8: Overhead for DSR.

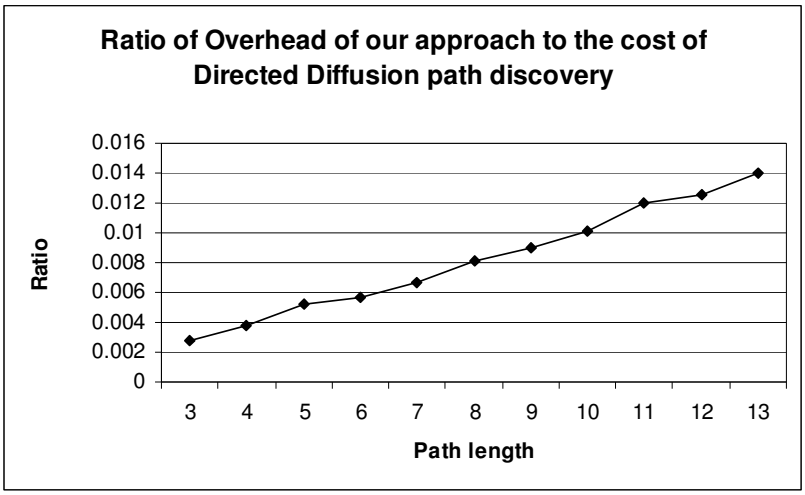


Figure 5.9: Overhead for Directed Diffusion.

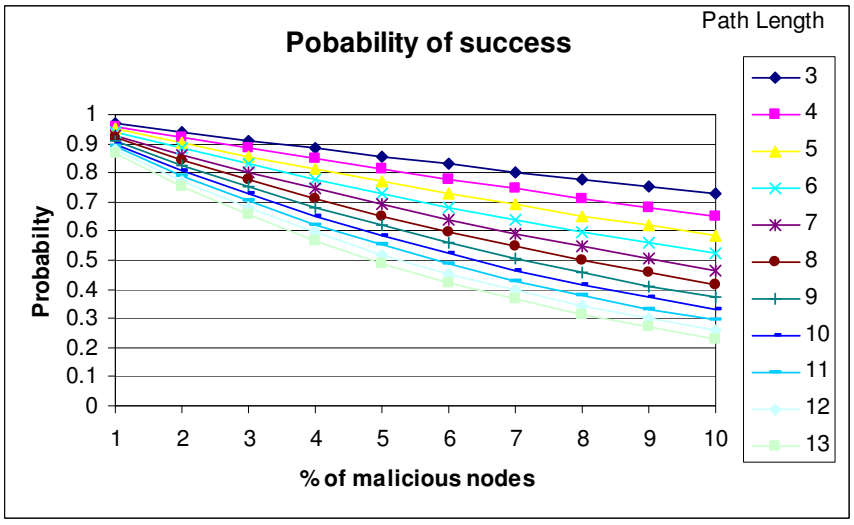


Figure 5.10: Probability of success of DPDSN (for a path length 3 to 13 for a network of 100 nodes in the presence of 1 to 10 percent of packet-dropping nodes).

5.4 Summary

Recent techniques for detection of packet-dropping nodes in ad hoc networks incur heavy costs and are not suitable for resource-constrained WSNs. DPDSN detects whether a *path* is dropping packets. Overhead for a two-dimensional grid of N nodes is $O(\sqrt{N})$ packets, where N is the number of nodes. The cost is independent of

the number of packets transmitted and the number of packet-dropping attacks being detected.

DPDSN incurs no cost for the response following the detection of a packet-dropping attack because the alternate path, established during route discovery, is ready for the response. The cost of finding an alternate path is the overhead paid during the route discovery, as shown in figure 5.8 and 5.9. We have shown that DPDSN works for DSR and Directed Diffusion.

Our simulations show that the overhead of DPDSN for a path of length 3 to 13, is approximately 0.6% to 2.6% of the energy the network consumes on DSR path discovery if the DSR protocol is used for routing. Similar overhead for Directed Diffusion is approximately 0.3% to 1.4%.

Results of this chapter appear in [59].

CHAPTER 6

DETECTION OF SINKHOLE, HELLO FLOOD AND EXHAUSTION

Abstract

Our solutions for detecting sinkhole, exhaustion and HELLO flood are based on statistical information about the traffic which is periodic and follows specific communication patterns. Solutions take into consideration important WSN characteristics like coverage, connectivity, data aggregation and communication patterns. We analyze the overhead of these techniques.

6.1 Introduction

Sinkhole and exhaustion deplete WSN nodes of energy. We assume that the attacker captures sensor node and alters its code to pose attacks. Adversary poses a HELLO flood attack by adding a device with more powerful radio.

6.2 Detection of Sinkhole

Recall from chapter 1 that WSN has specific communication patterns [1]. In most of the applications, nodes report their observations to the clusterhead which aggregates the result and forwards it to the base station. When nodes are static, they know their neighbors and they know that data flows towards the base station. Since nodes are stationary, they can easily detect when someone tries to attract data in the opposite direction. Therefore it is very difficult for an attacker to pose the sinkhole attack as in [15]. Based on our WSN model, it is safe to assume that for a given finite amount of time, the network topology remains the same i.e., the paths that are used to forward data remain same.

We assume that a sensor node is physically compromised and its code has been altered so that it can pose sinkhole attack. Adversary tries to attract traffic towards itself when new paths are set up to send data to the base station.

We divide the square area into eccentric strips of width R as shown in figure 3.10(a). Base station is located at (0, W). This helps us count the approximate number of hops for aggregators located in the strip to reach the base station. Let $W=cR$ for some constant c. We call an area "k-th strip" if all nodes in that area are not farther than kR and not closer than $(k-1)R$ for $k \geq 1$. So nodes in strip k are not less than k hops away from the base station. If a node in the k^{th} strip (for $k > 2$) is a sinkhole then its predecessor node in strip (k+1) will experience more traffic than normal, whereas its successor nodes in strip (k-1) will experience less traffic than normal. Also the base station receives less number of packets per iteration. We use this observation to detect sinkhole attack. Whenever nodes experience unusual traffic, they report the suspect to the base station. Based on this, base station detects whether some node is a sinkhole. Next we formulate the approximate number of packets each clustehead generates in one iteration. Please recall from lemma 3.3 that the area of strip k,

$$S_k = \frac{\pi(2k-1)R^2}{4} \qquad \text{for } 0 < k \leq c$$

$$= \frac{(2k-1)R^2\theta}{2} - 2A_1 - 2A_2 \qquad \text{for } k > c$$

where $\theta = \frac{\pi}{2} - 2\cos^{-1}\left(\frac{W}{(k-1)R}\right)$, $\alpha = \cos^{-1}\left(\frac{W}{kR}\right) - \cos^{-1}\left(\frac{W}{(k-1)R}\right)$,

$A_1 = \frac{k^2R^2\alpha}{2} - k^2R^2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right)$ and

$$A_2 = \left(\frac{-W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k}\right)\right)} + \frac{W}{\tan\left(\pi - \sin^{-1}\left(\frac{c}{k-1}\right)\right)} \right) . \frac{R}{2} . \sin\left(\frac{\pi}{4} + \frac{\theta}{2}\right).$$

Let C_i be the number of clusterheads in strip i. $C_i = \frac{2S_i}{\pi r^2}$ for $1 \leq i \leq K$ for $K = \sqrt{2}.c$.

Nodes in strip j receive $\sum_{i=j+1}^{K} C_i$ packets from nodes in strip j+1. Each cluster in strip j receive approximately $(\sum_{i=j+1}^{K} C_i)/C_j$ packets. Nodes in strip j send $\sum_{i=j}^{K} C_i$ packets to nodes in strip j-1. Each cluster in strip j sends approximately $(\sum_{i=j}^{K} C_i)/C_j$ packets. When a node in strip j is a sinkhole, then its successor in strip j-1 receives no packets or lesser number of packets, even though its predecessor in strip j+1 has sent $\sum_{i=j+1}^{K} C_i$ packets. When such a behavior is observed by predecessor and successor nodes, then the base station concludes that the node suspected by them is a sinkhole. We have just established the following result.

Theorem 6.1: Let nodes u, v, and w be in strip k-1, k and k+1 respectively. w has sent $\sum_{i=k+1}^{K} C_i$ packets to v, but u does not receive any packet from v then u guesses that v is a sinkhole. u sends encrypted message to the base station stating that v is a sinkhole. Base station verifies this claim by the fact that it received lesser number of packets than expected from strip k.■

In order to detect sinkhole a node needs the value of k (the strip number in which it is located) to count the estimated number of packets it should receive from its

predecessor. The process of computing the value of k at each node can be embedded in the original path discovery as shown in figure 6.1. The base station broadcasts a packet with integer value h=1 in the packet. When a node receives multiple packets with different h values it chooses the packet with minimum value of h. The minimum value of h is the value of k for the node. It then increments the value of h and broadcasts a packet (figure 6.1). Eventually all nodes compute its value of k (figure 6.2).

When nodes include location and the strip number in packets, then the base station can also detect location of node posing sinkhole attack. Base station estimates approximately $(\sum_{i=j}^{K} C_i)/C_j$ packets from the clusterhead in strip j during each iteration. The base station can even find the location of sinkhole from the observations of successor and predecessor nodes. This in turn will help intrusion response mechanism. However our goal in this chapter is to only detect intrusions.

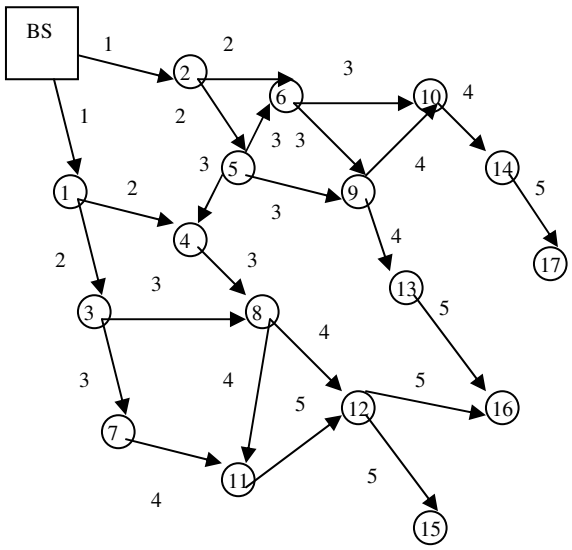


Figure 6.1: The h value is displayed for each arrow.

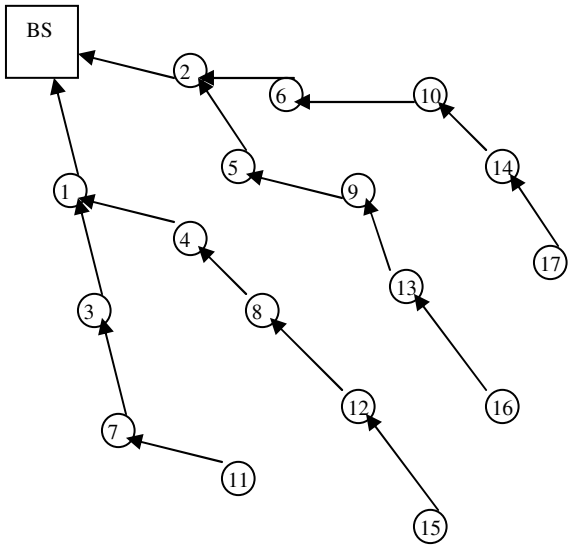


Figure 6.2: In the end, base station sets up paths with the nodes.

6.3 Detection of Exhaustion

Attackers can induce battery exhaustion in sensor nodes—for example, by sending a sustained series of useless communications that the targeted nodes will expend energy processing and may also forward to other nodes [44]. The expected number of packets from the predecessor depends on the strip in which a node is located. Clusterhead in strip j receives approximately $\sum_{i=j+1}^{K} C_i / C_j$ packets from nodes in strip j+1. When it receives more packets than expected then it guesses that its neighbor is performing exhaustion attack. It informs the base station about it. Base station expects approximately $\sum_{i=j+1}^{K} C_i$ packets from nodes in strip j+1. If a node in strip j+1 is performing exhaustion attack then base station receives more than $\sum_{i=j+1}^{K} C_i$ packets from nodes in strip j+1. Thus base station can verify that a node in strip j is

indeed performing exhaustion attack. The cost of detection is minimal since nodes shave to calculate only the statistical information.

6.4 Detection of HELLO Flood

Many protocols require nodes to broadcast a HELLO packets to announce themselves to their neighbors, and a node receiving such a packet may assume that it is within radio range of the sender. This assumption may be false: a laptop-class attacker broadcasting routing or other information with large enough transmission power could convince every node in the network that the adversary is its neighbor [1].

We assume that nodes send a HELLO packet to discover their neighbors after deployment. Attacker must send a HELLO packet with *id* from 1 to *N*. If it sends HELLO packet with some other *id* then it is an anomaly and the presence of an attacker can be detected easily. We assume that the attacker has transmission range greater than 2*r*. We know that attacker assumes *id X* such that $X \in \{1, 2, 3, \ldots, N\}$. Attacker may or may not disable actual node *X*.

If attacker does not disable node *X*, and if node *X* receives the HELLO packet with sender *id* field set to "*X*" then it notices that an attacker is masquerading its *id* and informs about this anomaly to the base station. After receiving HELLO packets, nodes send the encrypted packet containing the list of *id*s of nodes that sent HELLO packets. From these packets base station derives the number of nodes any node can reach. Assuming uniform placement of nodes, every node should be reachable to no more than $\frac{\pi r^2 N}{W^2}$ nodes. If any node reaches approximately at least $\frac{4\pi r^2 N}{W^2}$ nodes then there is a possibility that the node is performing the HELLO flood attack. Base station can detect this anomaly easily and detect the node causing the HELLO flood attack.

6.5 Detection of Packet-dropping

The mechanism proposed for detecting sinkhole can be used for detecting packet dropping.

6.6 Results

We use ns2 [46] to simulate WSN. Simulation parameters are shown in figure 6.3. We placed 100 nodes randomly in a square area of length W=100m. Nodes have transmission range of r=10m. Nodes that are in communication range form clusters. We found that for each iteration, a node consumes an average of 1.88 J of energy. When a base station broadcasts a packet, a node consumes an average of 0.631 J of energy. When a base station eliminates a node, each node consumes 1.272 J of energy. The same amount of energy is consumed by every node when base station adds new node to the network. When a node reports an anomaly, the nodes that forward this message, each consumes maximum of 1.93 J of energy. The effect on the network, of reporting anomalies, depends on how far the node is located from the base station and the number of anomalies detected. In some cases base station uses multiple anomaly observations to detect an attack.

W=100
r=10
N=100
Packet size=100 bytes
Energy to send a packet, E_s =1.38 J
Energy to receive a packet, E_r =0.549 J
RC5 encryption/decryption energy per packet= 0.076 mJ.
SHA1 One-way function per packet=0.065 mJ.
PWR, Initial node energy=1000 J

Fig 6.3: Simulation parameters.

Figure 6.4 shows the average energy a node consumes on various activities, whereas figure 6.5 shows the maximum energy any node consumes while detecting different attacks. Figure 6.6 shows the energy consumed for detecting an attack compared to initial node energy, PWR=1000 J. In case of Sybil, we give two suspected nodes 0.5 second time to transmit packets to two nodes. Each of the nodes that perform a test consume maximum of 22 J of energy. Detection of sinkhole incurs cost of computing the strip number k at each node and the cost of sending anomalies by successor and predecessor of the suspected sinkhole. Exhaustion consumes minimal energy because it only requires calculating the statistical information about the expected number of packets.

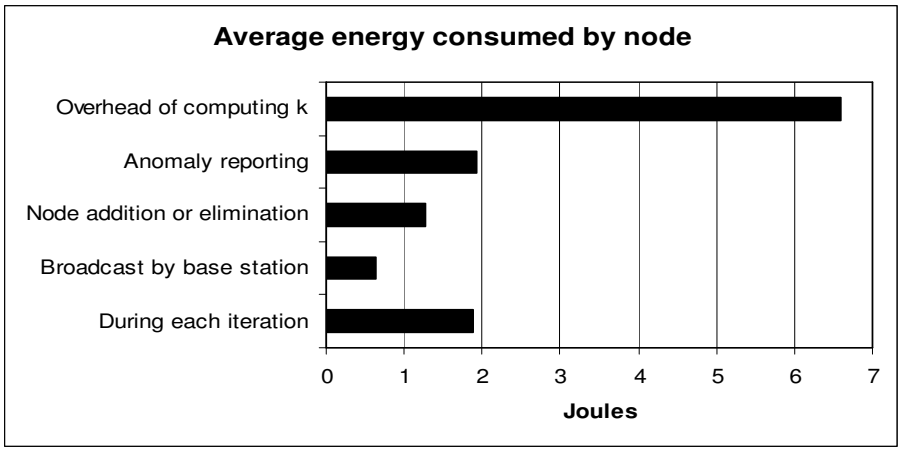


Fig 6.4: Average energy consumed by node during different activities.

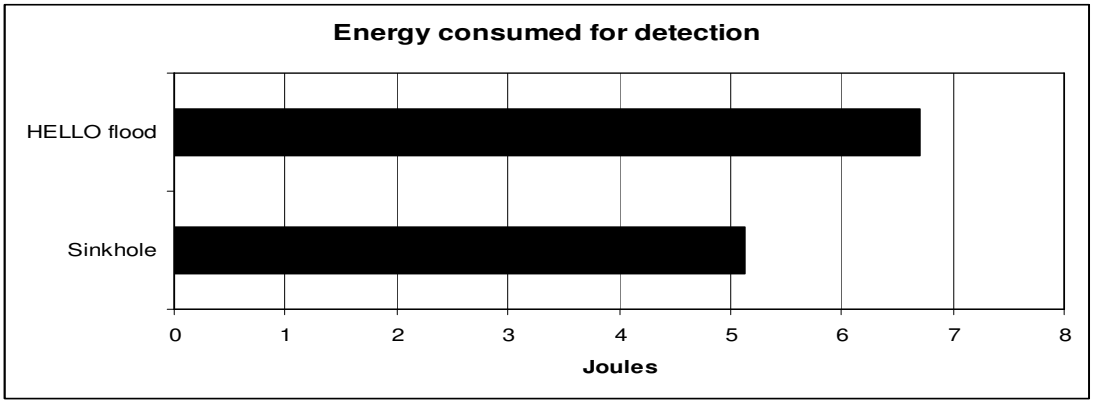


Fig 6.5: Maximum energy consumed by node while trying to detect different attacks.

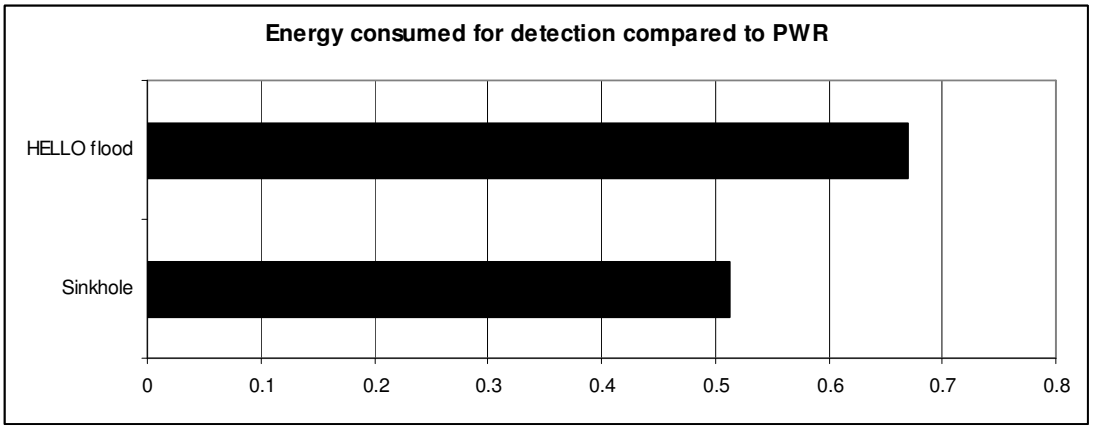


Fig 6.6: Energy consumed for detecting an attack compared to initial energy of nodes, PWR=1000 Joules.

6.7 Summary

We proposed lightweight techniques to detect attacks on WSN such as sinkhole, exhaustion and HELLO flood. The techniques are based on statistical information about traffic. The technique proposed for detecting sinkhole also detects packet dropping. Sinkhole detection technique depends on estimating the number of packets a node should receive depending upon its location. The solutions to detect attacks mentioned in this chapter form the second line of defense complimenting the first line of defense provided by cryptography based security primitives that prevent attacks. Results in this chapter appear in [56].

CHAPTER 7

DETECTION OF DATA FORGING BY CLUSTERHEADS

Abstract

In this chapter we describe mutual guarding based solution to detect data forging by an aggregator. This mechanism costs minimal overhead because it uses only the passive listening on the communication.

7.1 Introduction

We assume that the attacking node (a sensor or an aggregator) is physically compromised and its code is modified to forge data. By data forging we mean maliciously changing sensed data and reporting it. We are not addressing the issue of detecting breach of packet integrity here.

7.2 Solution to Detect Data Forging by an Aggregator

There can be two reasons [42] for incorrect data or results:

(i) Misbehaving sensor or aggregator and

(ii) Faulty sensor or aggregator.

We eliminate the possibility of the second case as explained next. Aggregator announces the *id* of the faulty sensor and ignores its readings. By doing this it assures and informs other nodes about its action of ignoring the readings from faulty sensor.

Since the medium is wireless, nodes overhear packets that are not destined to them. So they can snoop on the communication between their neighbors. We use overhearing to find anomalies and detect data forging. Nodes use the following to detect anomalies.

(i) Overheard data packets from other nodes and

(ii) Overheard results reported by the aggregator.

From anomalies, the nodes guess whether the aggregator is cheating by using the lightweight analysis module (as shown in figure 7.1). Our approach is different from Secure Information Aggregation [42]. Nodes use explicit messages to verify the validity of data in [42], whereas, in our proposed approach, nodes overhear the communication of their neighbors. This approach incurs very little cost of passive listening. It does not need explicit transmission of packets.

The lightweight analysis module is different for different functions such as AVERAGE, MIN, MAX etc. As an example, below we discuss the lightweight analysis module for calculating AVERAGE of the data sensed by sensor nodes.

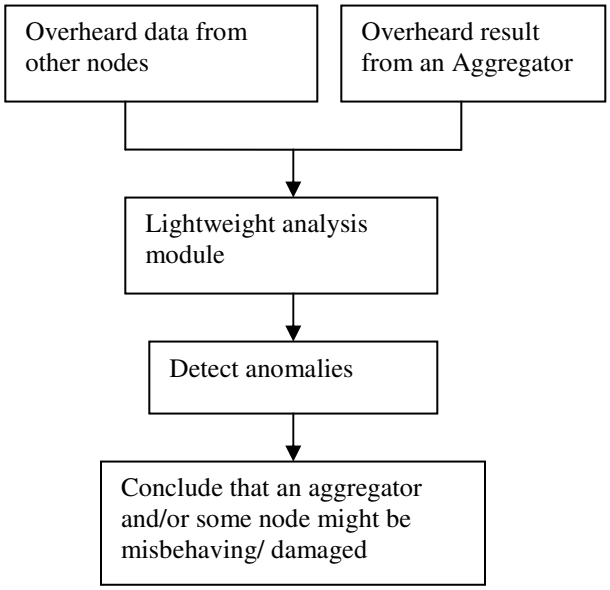


Figure 7.1: General idea behind data forging detection technique at a monitoring node.

Let us assume that there are $m > 1$ nodes in the cluster and a node can overhear transmission from mp nodes for $0 \leq p \leq 1$. Let $R_1, R_2, R_3, \ldots, R_m$ be the readings of sensor nodes 1, 2, 3, …. , m respectively. Let R_{min} and R_{max} be the minimum and maximum possible values of the readings. Aggregator ignores sensor nodes with

readings that are lesser than R_{min} and greater than R_{max}. Let R_A be the result the aggregator calculates and the node overhears. Let R_{mp} be the sum of sensor readings overheard by a node. Then, if $\frac{R_{mp}+(1-p)mR_{\min}}{m} > R_A$ or $R_A > \frac{R_{mp}+(1-p)mR_{\max}}{m}$ then, it is an anomaly and we conclude that an aggregator is cheating. Nodes closest to the aggregator can overhear the maximum number of nodes and are best suited to do the job of detecting whether an aggregator is cheating. It is possible that there may not be even a single node that overhears all the packets sent to the aggregator. Therefore nodes are not 100% confident about their guess that the aggregator is cheating. Even though sensor nodes are not 100% confident about their guess, they do not generate any false alarms. Next we formulate the confidence, nodes have in their guess as a function of $\frac{l}{r}$ where l is the distance between the clusterhead and the node and r is the communication range as defined earlier. Since nodes are distributed evenly, the confidence is proportional to the distance of a node from the aggregator.

Theorem 7.1: A node detects cheating by the clusterhead with $\frac{200}{\pi r^4}\left(\cos^{-1}\left(\frac{l}{2r}\right)-\frac{l}{2r}\sqrt{1-\left(\frac{l}{2r}\right)^2}\right)$% confidence.

Proof: Monitoring node can monitor readings from nodes that are located in the common area. The common area (marked by squares), $A_{common}=\frac{2}{r^2}\left(\cos^{-1}\left(\frac{l}{2r}\right)-\frac{l}{2r}\sqrt{1-\left(\frac{l}{2r}\right)^2}\right)$. Aggregator receives data from sensor nodes that are placed in πr^2 area. Therefore the monitoring node has $\frac{A_{common}}{\pi r^2}*100\,\%$ confidence in its guess. The theorem follows.■

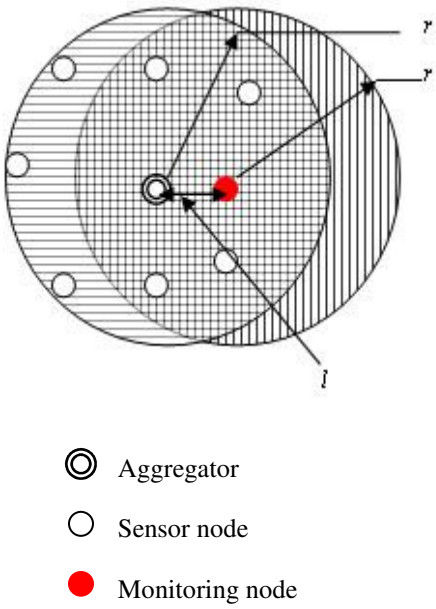


Fig 7.2: The closer the monitoring node is to the aggregator the more readings it can overhear.

For example, when $\frac{l}{r}$=0 and r≠0 then confidence is 100%. Whereas when l=r then confidence is 39%.

When a node overhears data value that is not in the range then they expect aggregator to announce the *id* of the faulty or misbehaving sensor node. In this case, if the aggregator does not inform all the nodes in the cluster about the presence of faulty or malicious node then nodes assume that the aggregator is cheating. Similarly when aggregator accepts readings from faulty or misbehaving sensor nodes and other sensor node notices it then they inform the base station about this anomaly.

If nodes know their location *a priori*, then two or more nodes can collaborate and they can monitor the aggregator. Collaboration among nodes that overhear packets from disjoint sets of nodes increases the probability of detecting misbehaving aggregator. But there is a cost involved in this approach. The cost is the communication between collaborating nodes. Finding appropriate monitoring nodes

is a challenge for the base station and involves overhead. For nodes with isotropic antennae we need six nodes placed around an aggregator.

7.3 Results

The proposed solution uses only passive listening; hence the overhead of detection is negligible. The cost of informing an anomaly to the base station is not included here.

Figure 7.3 shows the relationship between confidence a node has in its guess about the behavior of the aggregator vs. *l/r* where *l* is the distance between the node and the aggregator and *r* is the communication range (as defined in WSN model in section 5.1) of nodes including that of the aggregator.

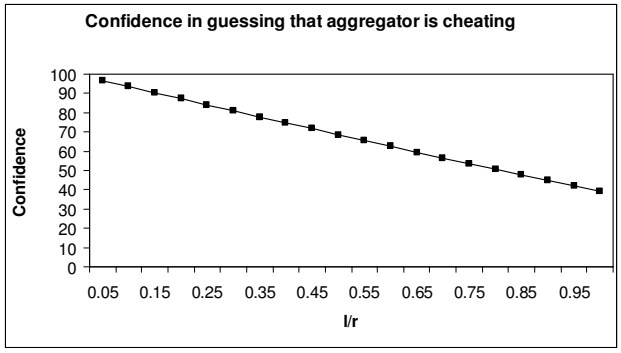


Fig 7.3: Confidence in detecting cheating by an aggregator – numerical result of theorem 7.1.

7.4 Summary

We propose a lightweight overhearing based mechanism to detect data forging by an aggregator. This mechanism involves minimal overhead of passive listening. Results of this chapter appear in [56].

CHAPTER 8

INFORMATION AUTHENTICATION BY DETECTION OF INVALID SOURCE OF INFORMATION (IASN)

Abstract

We propose lightweight methods to detect invalid source of information for WSN. The main idea is to reuse the already available system information generated by routing protocols for detecting anomalies.

8.1 Introduction

We propose the use of forwarding tables generated by routing protocols for detecting invalid source of information. This detection ensures *information authentication for sensor networks* hence we denote it as IASN in the rest of this chapter. We analyze and extend IASN in this chapter. We show how it works with routing protocols like DSR [36], DSDV [9] and Directed Diffusion. We then extend it where a node running IASN keeps track of neighbors that are k hop away for $k > 1$.

We term high-level data as *information*. The main purpose of sensor networks is to sense some environmental variables and send readings periodically to a base station or send readings whenever someone demands them. Since multiple sensors are deployed to sense some environmental variables it is expected that they collaborate among themselves to generate meaningful information (if sensors are deployed to sense fire in woods, then detection of fire is meaningful information).

IASN detect intrusions that can occur while information is on route from source to destination. IASN keep track of neighbors and the type of information expected from them. Upon information arrival it matches the information against the neighbor and verifies it. If information is not supposed to come from a certain node, it guesses

that it is an attempt to infuse malicious packets. We show that IASN works with source initiated, receiver initiated or table driven routing protocols. We tested our protocol IASN with DSR and DSDV.

8.2 IASN

Next we summarize IASN. Once a path is established by a routing protocol and nodes know what information to expect from each of the neighbors, they can detect an anomaly if they receive that information from different neighbor. Suppose node *E* of figure 8.1 sends a packet to node *A* via node *B* containing sensed temperature ($INFO_3$) data, then node *A* expects temperature data from node *B* and node *B* expects temperature data from node *E*. If nodes *A* and *B* receive the temperature data from any other nodes except nodes *B* and *E*, respectively, then they can detect and filter (drop) forged packets. In general $INFO_i$ is meaningful information like an answer (for example sensed temperature of some area at some particular time) to some query etc. Let $INFO_1$, $INFO_2$, $INFO_3$, …, $INFO_p$ indicate the *p information*s that a node receives from neighbors $N_1, N_2, N_3, ..., N_p$ respectively.

Every node running IASN can maintain an anomaly detection table (ADT) containing the list of neighbors that may forward some particular *information* to that node. IASN protocol then detects and drops forged packets by comparing the *information* in the packet and the source of the packet against the entries in the ADT. Consider the WSN scenario of figure 8.1 in more detail. Suppose nodes *G*, *H*, and *E* have established paths reaching node *A* to send $INFO_1$, $INFO_2$, and $INFO_3$ respectively. Node *A* gets $INFO_1$ from node *C*, $INFO_2$ from node *D* and $INFO_3$ from

B. Node *A* can maintain an ADT. Now suppose there is an adversary *X*, which sends a forged packet containing $INFO_1$ to node *A* via node *D* (see figure 8.2).

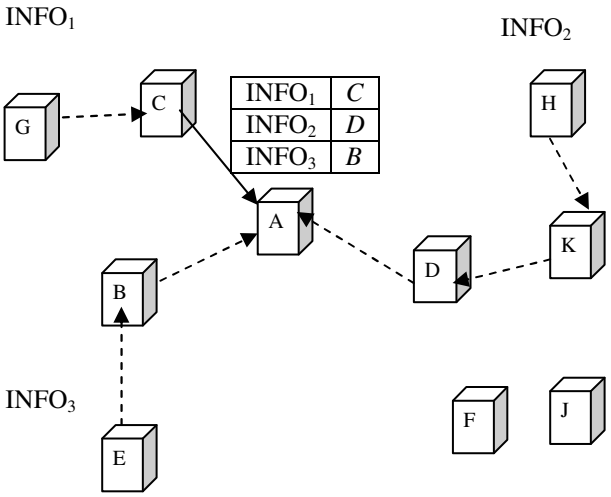


Figure 8.1: ADT for node *A*. Node *A* expects $INFO_1$, $INFO_2$ and $INFO_3$ from neighbors *C*, *D* and *B*, respectively.

If node *A* gets this packet from node *B* or *D* then it can detect that the packet is forged because it expects $INFO_1$ only from node *C*. If node *A* keeps information about route updates and keeps the ADT consistent with the current routing paths then it can very easily detect packet spoofing. Note that an adversary *X* can spoof a packet with $INFO_2$ via node *D*. However, if node *D* also maintains ADT and runs IASN protocol, then this forged packet can be detected at node *D*. Figure 8.3 explains that case. Designers can thus use the system information to detect intrusions.

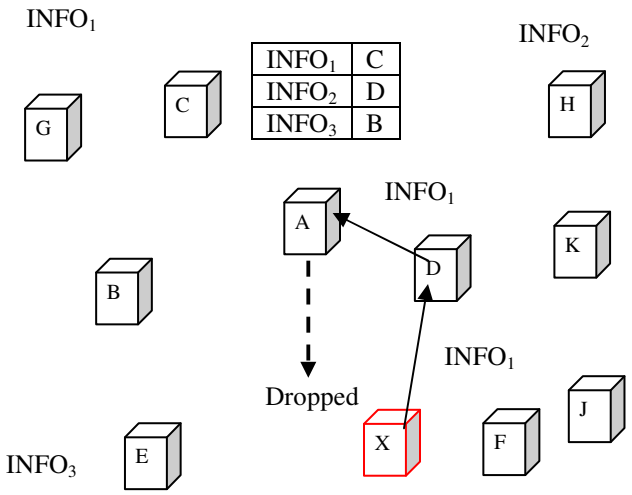


Figure 8.2: A detects forged packet from adversary X because it expects $INFO_1$ only from C.

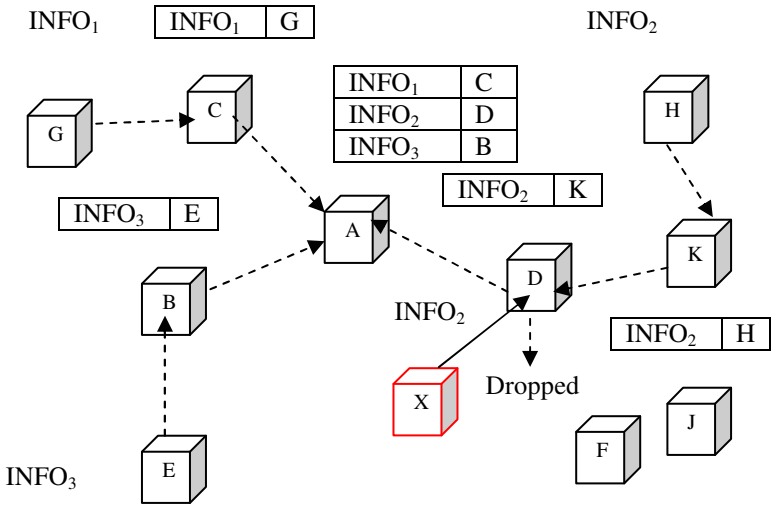


Figure 8.3: *X* sends $INFO_2$ to node *D*, which drops it.

In figure 8.3, nodes *G*, *H*, *E*, *F* and *J* have empty ADTs, as they are not receiving any information. Whereas *A*, *B*, *C*, *K* and *D* have some entries in their tables. In this example, if all nodes run IASN and if we assume that nodes do not masquerade (detecting masquerade without an explicit node authentication mechanism is a challenge) then all the forged packets will be detected and dropped. In the previous example, a node receives INFO of a certain type from a single neighbor. The ADT

can be easily extended to accommodate the situations where (i) More than one neighbors are allowed to forward the same INFO, or (ii) Multiple INFOs are forwarded by a neighbor. Efficient representation of ADT should be used to optimize the performance of IASN with respect to time and space requirements.

It is easy to see that the storage overhead of ADT is proportional to the number of neighbors. ADT can be easily derived and maintained from the underlying data dissemination mechanisms or the routing protocols. The forwarding-tables or next-hop information of the routing protocols can be used to build the ADT. Furthermore route-update messages can be used to keep ADT in concurrence with the routing path changes. Figure 8.4 shows the updated ADT after a $H \rightarrow D$ path changes from figure 8.3. This method of detecting anomaly intrusions is thus lightweight because it uses existing routing data.

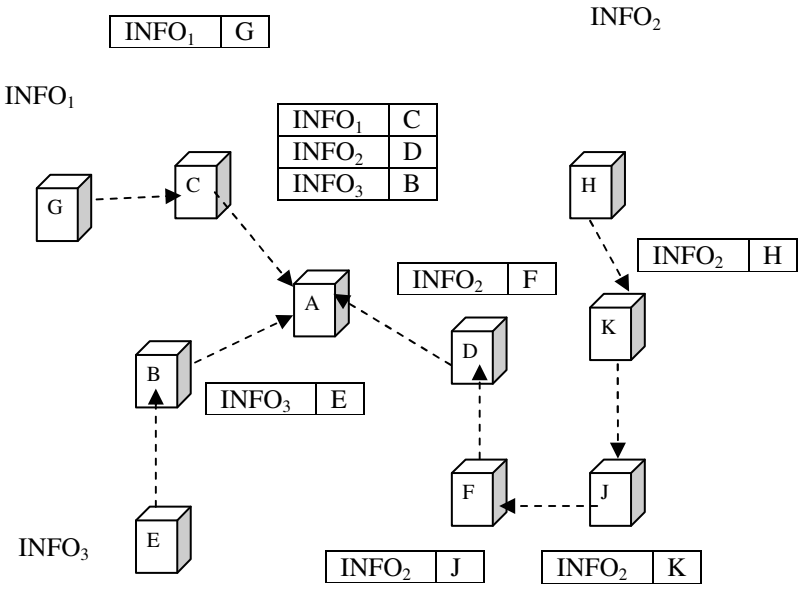


Figure 8.4: ADTs at nodes after change in route form *H* to *A*.

8.3 Constructing ADTs for Source Initiated on Demand ad hoc Routing Protocols (DSR)

DSR is an on demand routing protocol. It has *route request* and *route reply* phases. Initially a source floods a route request packet as shown in figure 8.5. A route reply is generated either by the destination or an intermediate node, which contains an unexpired route to a destination in its route cache [47]. In route reply phase when packet is coming back to the source, ADTs can be built at all the nodes that are on the path as shown in figure 8.6. If that route is not needed anymore then a Cancel message has to be sent by a source to all the nodes on the route to delete ADTs.

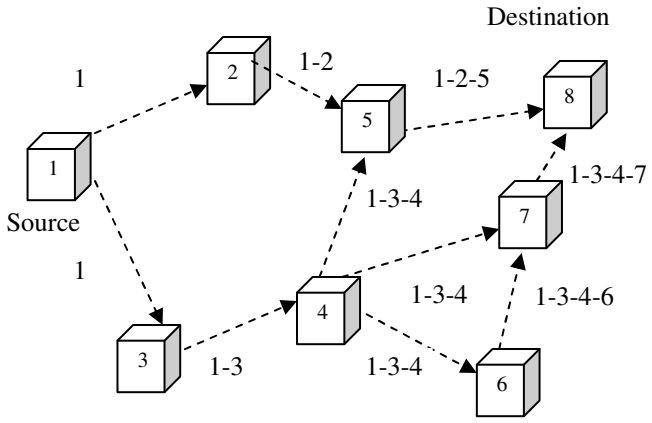


Figure 8.5: Node 1 floods route request packet.

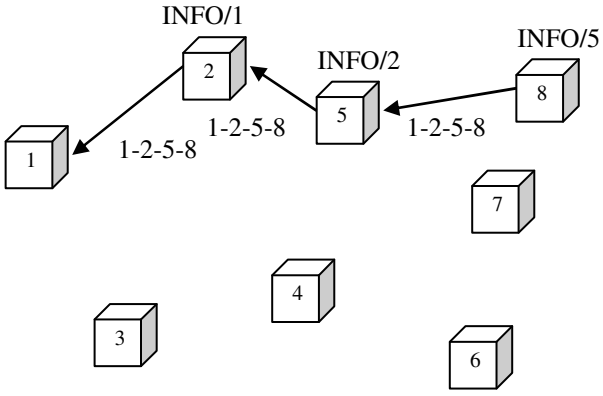


Figure 8.6: ADTs built during route reply phase.

Let the cost of writing one byte into a packet be w, the cost of reading one byte from a packet be r, the cost of sending a packet be s and the cost of receiving a packet be v. Let n be the number of nodes, m be the maximum number of neighbors any node has. Let p be the maximum length of any path. The approximate cost of route request and route reply is $n(w + s + mv)$ and $p(s + v)$ respectively. Therefore the cost of DSR path discovery is $n(w + s + mv) + p(s + v)$. The cost of constructing ADTs at the nodes is pr. Let us assume that q messages are sent once a path is discovered and ADTs are constructed. The cost of writing source-id in a packet is qpw, whereas the cost of reading source-id from a packet is qpr. Therefore the cost of checking whether the information is coming from an intended neighbor is $qpr + qpw$. The cost of Cancel message is $p(s+v)$. Therefore the overhead is $(qpr + qpw + pr+ p(s + v))$. The cost of message transmission over p hops is $qp(s+v)$. The ratio of overhead to the energy network consumes on DSR path discovery and transmission of packets using that path over some time period is $(qpr + qpw + pr+ p(s + v)) / (n(w + s + mv) + p(s + v) + qp(s+v))$. To get an idea of the above ratio empirically, consider a practical situation in which $n=q=100$, $m=6$, $s=v$, $r=w$ and $p=10$, the ratio is $(2010 + 20(s/r)) / (2720(s/r) + 100)$. The cost of sending a packet is much greater compared to reading a byte from a packet. If we assume s/r to be 100 and all p nodes run IASN, then they can detect all the forged packets at the cost of 1.4 per cent energy the network consumes on DSR path discovery and transmission of packets using that path over some time period.

8.4 Constructing ADTs for Directed Diffusion

Directed diffusion is a receiver-initiated protocol. In directed diffusion, destination (sink) floods the network in search of some data (called as "Interest") as shown by dotted arrows in figure 8.7. Whenever that message reaches source, it floods the network back as shown by solid arrows. Then it reinforces only one path to sink, which is used for all future communication (see figure 8.8). It is during the reinforcement phase that we can construct ADTs on nodes that are on the path. A Cancel message has to be sent in order to wipe out ADTs if that path is no longer valid. We analyze overhead of IASN similar to DSR. The ratio of overhead to the energy network consumes on path discovery and transmission of packets using that path over some time period is $(qpr + qpw + pr + p(s + v)) / (2n(s + mv) + p(s + v) + qp(s+v))$. For $n=q=100$, $m=6$, $r = w$, $s=v$, $p=10$ and $s/r= 100$, the above ratio is 0.011 which is very small. If all p nodes run IASN, then they can detect all the forged packets at the cost of 1.1 per cent energy the network consumes on path discovery using directed diffusion and transmission of packets using that path over some time period.

In DSDV every node maintains a routing table in which all the possible destinations and the number of hops to it are recorded (see table 8.1). Routing table updates are done in two ways: *full dump* and *incremental*. Full dump carries all the routing information whereas incremental carries only those updates, which have happened after the previous full dump. Table 8.2 is the advertised route table by node 4. In that table we can append a field that will tell neighbors what INFO they should

expect. When a route table is advertised, neighbors can construct their ADT (see figure 8.10). Thus constructing ADT at nodes running DSDV incurs a very little cost.

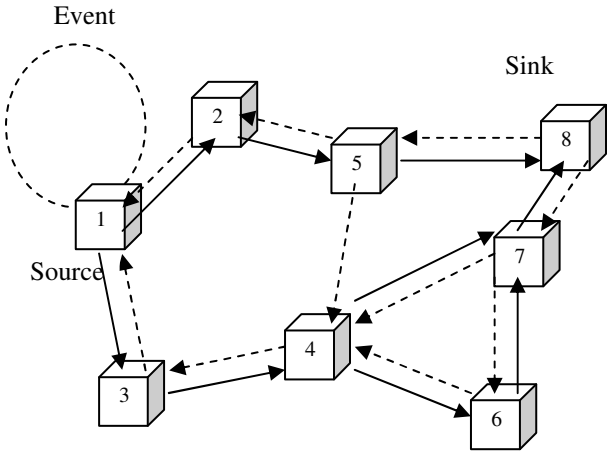


Figure 8.7: Sink floods interest (dotted arrows). Source replies with gradient message (solid arrows).

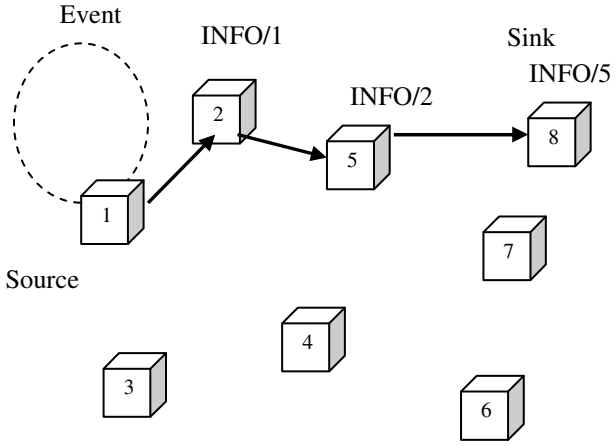


Figure 8.8: Source reinforces one route. ADTs can be built during reinforcement of a route.

8.5 Constructing ADTs for Table Driven Protocol (DSDV)

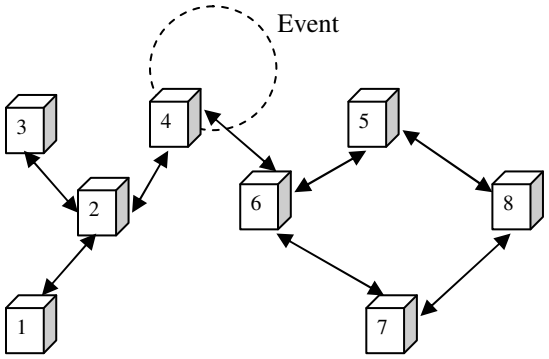


Figure 8.9: All nodes maintain routing tables as shown in table 1 for node 4 in DSDV ([9]).

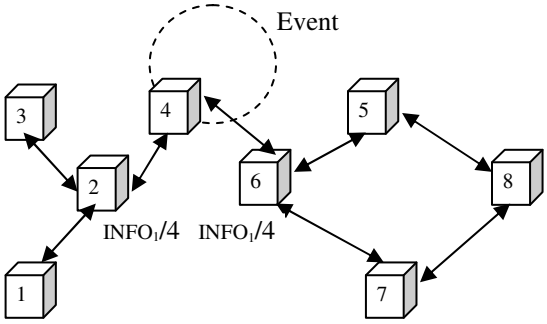


Figure 8.10: ADTs are built after node 4 advertises.

Destination	Next Hop	Metric	Sequence No.
1	2	2	S406_1
2	2	1	S128_2
3	2	2	S564_3
4	4	0	S710_4
5	6	2	S392_5
6	6	1	S076_6
7	6	2	S128_7
8	6	3	S050_8

Table 8.1: Forwarding table of node 4 for the network of figure 8.9 (modified from [9]).

Destination	Metric	Sequence No.	INFO
1	2	S406_1	
2	1	S128_2	$INFO_1$
3	2	S564_3	
4	0	S710_4	
5	2	S392_5	
6	1	S076_6	$INFO_1$
7	2	S128_7	
8	3	S050_8	

Table 8.2: Modified advertised route table of node 4 to construct ADTs (courtesy [9]).

8.6 Analysis of IASN

Let us consider a path from source *1* to destination *N* through nodes *2,3, …, N-1* as shown in figure 8.11.

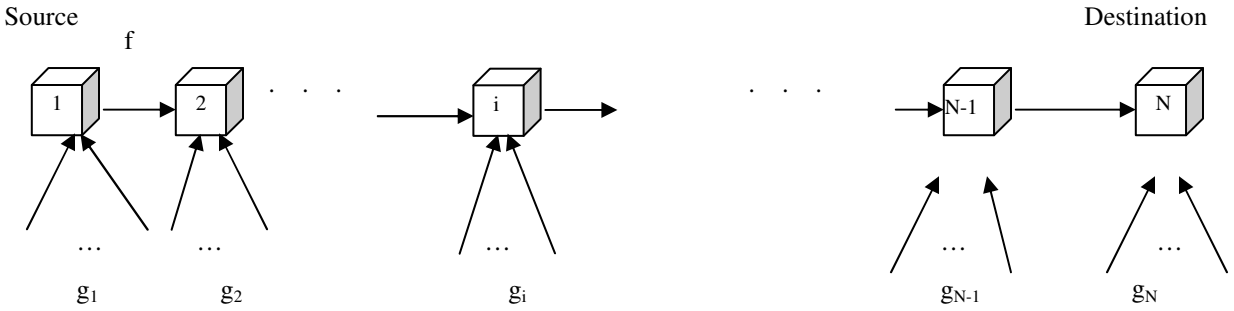


Figure 8.11: A route from 1 to N.

Node *1* is sending information, INFO to node *N*. Let *f* be the number of packets containing INFO sent by the source to node *2*. Let g_i be the number of packets with INFO forwarded by an adversary directly to node *i* or through neighbors of node *i* except its predecessor *i-1* to node *i* for $1 \leq i \leq N$. All g_1 packets will be dropped by source (node *1*) because source itself generates INFO. Let m_i be the number of

packets that reach node i for $1 \leq i \leq N$. m_1 is 0 because node 1 is the source of INFO and it does not accept packets containing INFO from any other node. We assume that all the packets an adversary sends are destined for the destination, i.e., node N. We also assume that all the nodes on the path forward packets to their successor, i.e., internal nodes do not cheat. For the purpose of this analysis we assume that adversary only tries to infuse INFO packets into the network. It does not masquerade as any other node.

Theorem 8.1: If node i runs IASN and no node $j < i$ runs IASN, then the probability of detecting a forged packet at node i is $g_i / \sum_{j=2}^{i} g_j$.

Proof: Out of nodes 2 through i only node i runs IASN. It receives g_i packets from its neighbors except its predecessor. It will detect and drop all those packets, as those are forged packets sent by adversary. The number of packets that reach node i is $m_i = f + \sum_{j=2}^{i} g_j$ .Only f valid packets are sent by the source. Therefore the total number of malicious packets that reach node i is $m_i - f$. Hence the probability of detecting a forged packet $= g_i /(m_i - f) = g_i / \sum_{j=2}^{i} g_j$. ∎

Theorem 8.2: Let $Q=\{X_1, X_2, \ldots, X_p\}$ be the set of nodes that run IASN such that and $X_j \leq i-1$ and $X_i \neq X_j$ for $i \neq j$ and $2 \leq j \leq p$. Then the probability of detecting a forged packet at node i that runs IASN is $g_i /(g_i + \sum_{\substack{j=2 \\ j \notin Q}}^{i-1} g_j)$.

Proof: Node i will drop g_i packets. The number of packets that reach node i is $m_i = f + g_i + \sum_{\substack{j=2 \\ j \notin Q}}^{i-1} g_j$. Out of these only f valid packets are sent by the source. Therefore the total number of malicious packets that reach node i is $m_i - f$. It follows that the probability of detecting a forged packet= $g_i / (m_i - f) = g_i / (g_i + \sum_{\substack{j=2 \\ j \notin Q}}^{i-1} g_j)$. ∎

Lemma 8.3: When all the nodes on the route (except source which does not have to run IASN) run IASN then the probability of detecting a forged packet is *1*.

Proof: This is a special case of theorem 2 with $p=N-1$ *and* $i=N$. Since all nodes $j < i$ run IASN, $\sum_{\substack{j=2 \\ j \notin Q}}^{i-1} g_j$ is *0*. The lemma now follows. ∎

Theorem 8.4: Let $Q=\{X_1, X_2,,\ldots, X_p\}$ for $1 < p \leq N$ be the set of nodes that run IASN on a route from node *1* to node *N*. Then the probability of detecting a forged packet destined for node *N* on the route is $(g_1 + \sum_{\substack{i=2 \\ i \in Q}}^{N} g_i) / \sum_{i=1}^{N} g_i$.

Proof: Node *1* will drop g_1 packets. Each of the node i that runs IASN will drop g_i packets. Therefore the total number of packets dropped is $(g_1 + \sum_{\substack{i=2 \\ i \in Q}}^{N} g_i)$. Adversary sends $\sum_{i=1}^{N} g_i$ packets. The probability of detecting a forged packet on the route from *1* to *N* is $(g_1 + \sum_{\substack{i=2 \\ i \in Q}}^{N} g_i) / \sum_{i=1}^{N} g_i$. ∎

8.7 Results

We simulated IASN protocol using ns2 [46]. For our simulations we considered an area whose boundary is defined as 100m x 100m. We tested IASN with two routing protocols DSDV and DSR. For each routing protocol we considered two types of topologies: fixed and random to simulate regular versus irregular (ad hoc) placements of sensor nodes. This results in four scenarios. In fixed topology 100 nodes are arranged in a 10 x 10 grid and are uniformly distributed over the area. In random topology, we placed the nodes randomly in the 100m x 100m area. We varied the number of nodes that are running IASN protocol. The nodes that run IASN protocol are selected randomly. This experiment was repeated for 1 to 100 nodes that run IASN protocol for all four scenarios. For all four scenarios an adversary is in one corner and a node under attack is in a diagonally opposite corner. The experimental results show that the number of packets that are detected increases as the number of nodes that run IASN protocol increased. This was observed for both the routing protocols DSDV and DSR.

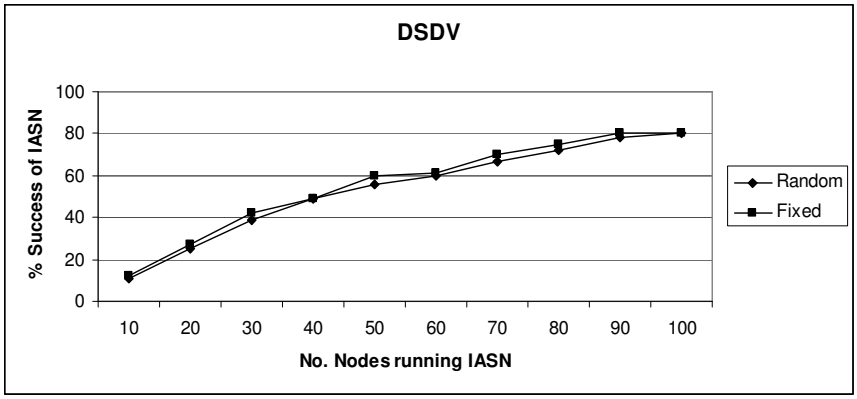


Figure 8.12: IASN with DSDV.

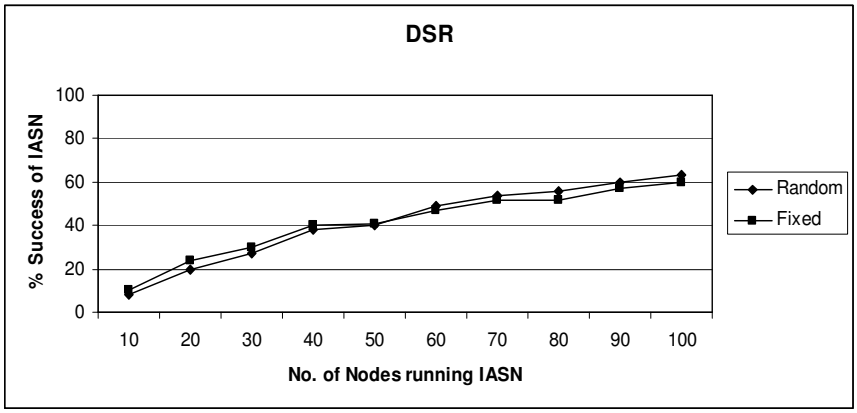


Figure 8.13: IASN with DSR.

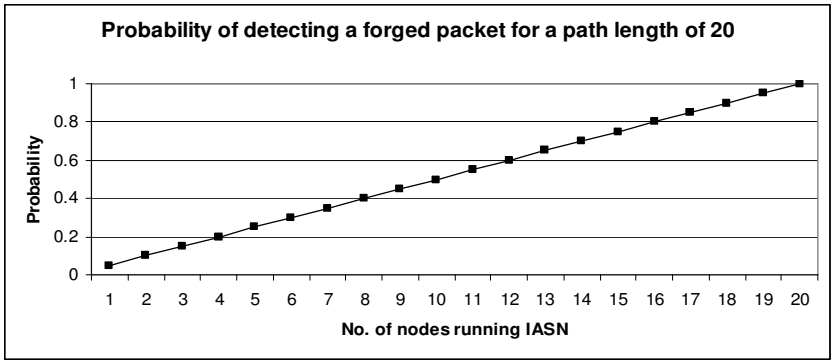


Figure 8.14: Probability of detecting a forged packet on a path.

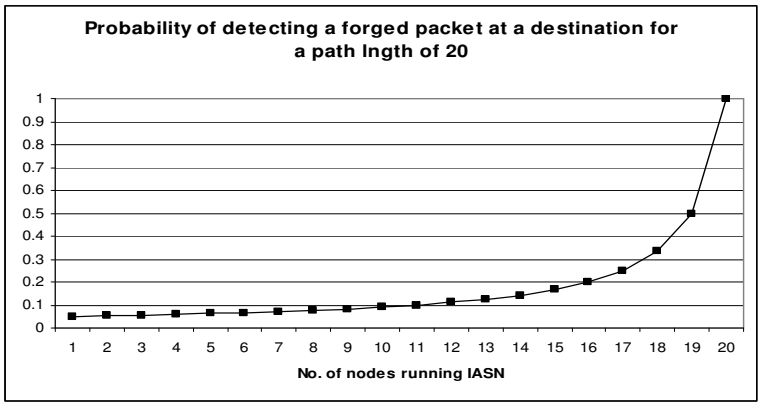


Figure 8.15: Probability of detecting a forged packet at a destination.

From our analytical analysis we found that the probability of detecting a forged packet on a path is proportional to the number of nodes that run IASN (figure 8.14

and theorem 4). The probability of detecting a forged packet at a destination is as shown in figure 8.15 (theorem 8.2).

8.8 Keeping Multi-hop Information

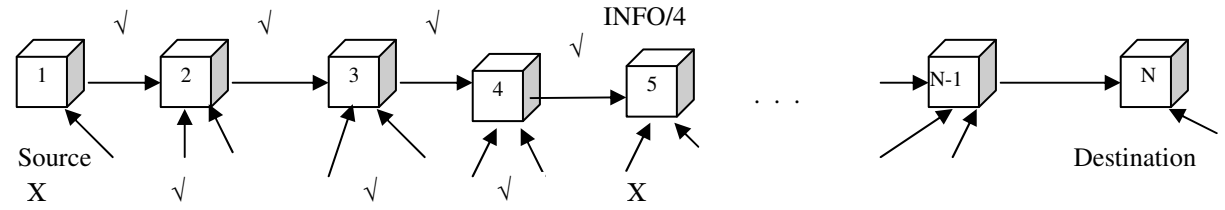
So far we were associating INFO with immediate neighbor. Let us consider a scenario where a node runs IASN and it keeps information about 2 predecessors that are forwarding INFO. In general if a node keeps track of *k* predecessors that are forwarding INFO then we call it a *k*-hop IASN. Figure 8.16 illustrates various scenarios of multi-hop IASN (i.e. *k*-hop IASN). We will see later that there is a tradeoff between k, storage space and number of nodes that run multihop IASN. In figure 8.16(a) node 5 runs 1-hop IASN and it drops the packets forwarded by neighbors of node 5 except its predecessor. In figure 8.16(b) node 5 runs 2-hop IASN. In this case it drops packets forwarded by neighbors of node 5 except its predecessor (node 4) and packets forwarded by neighbors of node 4 except its predecessor (node 3). In figure 8.16(c) nodes 4 and 5 run 2-hop IASN. In this case they detect spoofed packets forwarded by neighbors of node 3, 4 and 5 except their predecessors (2, 3 and 4 respectively). Note that even if we run 1-hop IASN at node 5 (and node 4 still runs 2-hop IASN) the same number of packets will be detected. This implies that running 2-hop IASN on successive nodes is same as running 2-hop IASN on the first node and 1-hop IASN on the second node. Figure 8.16(d) implies that running 2-hop IASN on alternate nodes is same as running 1-hop IASN on every node. But with this approach we have to send *id*s of 2 nodes in a packet. So we double the number of *id*s that we send in a packet. We also generate and store ADTs on only half of the nodes compared to a 1-hop approach. We have to update packet only on alternate nodes

instead of every node. The above approach can be easily generalized for k-hops. If nodes that are k-hop apart run k-hop IASN then it is same as running 1-hop IASN on all nodes.

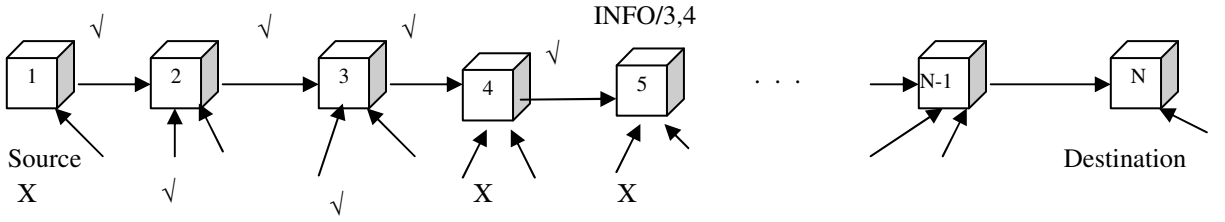
Theorem 8.5: If node i runs 2-hop IASN and no node $j < i$ runs IASN or 2-hop IASN, then the probability of detecting a forged packet at node i is $(g_i + g_{i-1}) / \sum_{j=2}^{i} g_j$ for $2 \le i \le N$.

Proof: Out of nodes *2* through i only node i runs 2-hop IASN. Node i receives g_i packets from its neighbors except its predecessor node *i-1*. Node *i-1* receives g_{i-1} packets from its neighbors except its predecessor *i-2*. Node i will detect and drop $(g_i + g_{i-1})$ forged packets. The number of packets received by node i is $f + \sum_{j=2}^{i} g_j \mid 2 \le i \le N$. Out of these packets only f valid packets are sent by the source.

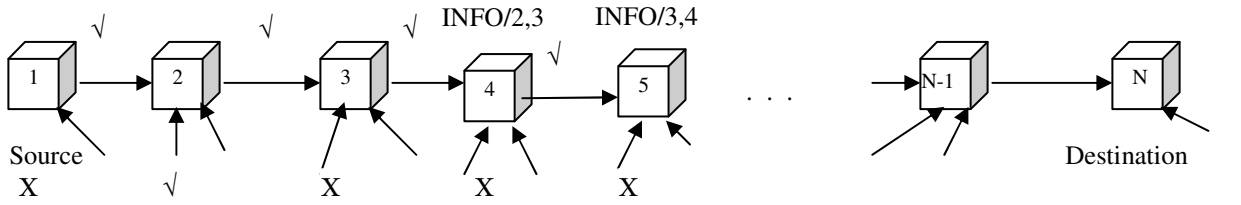
Therefore the number of forged packets that reach node i is $\sum_{j=2}^{i} g_j$. The probability of detection= $(g_i + g_{i-1}) / \sum_{j=2}^{i} g_j$. ∎



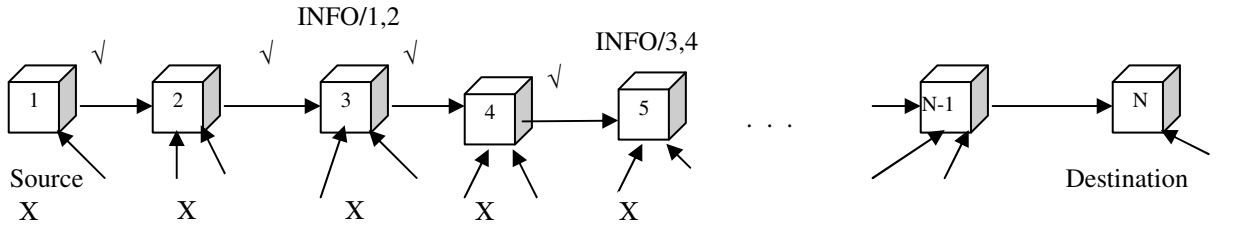
(a) Node 5 runs 1-hop IASN



(b) Node 5 runs 2-hop IASN



(c) Nodes 4 and 5 run 2-hop IASN



(d) Nodes 3 and 5 run 2-hop IASN

Figure 8.16: Multi-hop IASN.

Legend: √ are neighbors whose packets will be accepted by nodes running IASN and X are neighbors whose packets will be dropped.

8.9 Summary

IASN uses already existing forwarding tables generated by routing protocols for detecting invalid source of information. We have shown that it can be used for source initiated routing protocols, table driven routing protocols and data dissemination

mechanisms like directed diffusion. The probability of detection increases linearly with the number of nodes running IASN. Running *k*-hop IASN on nodes that are *k* hops apart is same as running IASN on all nodes with 1-hop information. We have shown that ADTs can be built while discovering routes for source initiated routing protocols, table driven routing protocols as well as data dissemination mechanisms like directed diffusion. The storage overhead of ADT is proportional to the number of neighbors. Approximate overhead of IASN is 1.4 percent of the energy network consumes on DSR path discovery and transmission of packets using that path subsequently. Similar overhead is 1.1 percent for directed diffusion. Results of this chapter appear in [40].

CHAPTER 9

CONCLUSION

We identified challenges in intrusion detection for WSN. We found that masquerade/Sybil and packet dropping are the precursors for many other dangerous attacks. Hence high priority should be given in detecting these basic attacks. We proposed a intrusion detection techniques for detecting most dangerous attacks like masquerade, Sybil, packet dropping, sinkhole, HELLO flood and data foging by an aggregator.

MG method for detecting masquerade/Sybil is based on overhearing the communication of immediate neighbors. This is a novel and fundamental contribution and can be extended for any wireless network. SRP method verifies the number of packets sent and received from nodes based on their *id*s. Both methods are independent and compliment each other in preventing attacks. MG method fails to protect nodes that are one hop away from the border or when attacker has shorter communication range than sensor nodes. SRP overcomes these drawbacks at a reasonable cost. MG method incurs insignificant overhead as it uses only passive listening. SRP decreases network lifetime by only 1% when it is run after 30 iterations (cycle during which all nodes send data to the base station). Overhead of SRP is minimal. The probability of success is very high for SRP.

We propose a technique to detect packet forging and sinkhole which estimates the number of packets a node should receive/send from/to its neighbors. Estimating the number of packets is possible because sensor nodes send data periodically to the base station. The overhead of computing the estimated number of packets at each node is

6.6 J per node when energy for sending a packet is 1.38 J and the energy for receiving a packet is 0.549 J. The overhead of detecting sinkhole is 5 J. DPDSN detects packet dropping paths and detects packet dropping nodes only if there is a need to do so. DPDSN works best for another type of WSN application – target tracking. We found that the overhead of DPDSN for a WSN of 100 nodes is approximately 2.6% of the energy the network consumes on DSR path discovery. Similar overhead for Directed Diffusion is approximately 1.4%.

We also propose technique based on overhearing for detecting data forging by sensor nodes and aggregator. Nodes report their observations with certain confidence. The confidence is more than 90% when the distance between node and the aggregator is less than 0.15r where r is the radius. Our work on detecting packet spoofing (IASN) is based on expecting certain kind of data from a neighbor. We embed the process of mapping neighbors to data in the routing protocol itself. The overhead is approximately 1.4% of the energy network consumes on DSR path discovery and transmission of packets using that path subsequently. Similar overhead is 1.1% for directed diffusion.

We also propose use of μTESLA based mechanism that allows sensor nodes to send anomalies or information about detected attacks/attackers to the base station.

We propose solutions to detect some of the most important attacks. Our solutions are localized (mostly overhearing communication in one-hop range, sharing information with neighbors or using statistical information based). They involve minimum overhead in terms of communication and hence are lightweight. We take into consideration important WSN characteristics like coverage, connectivity, data

aggregation, communication patterns and periodic traffic. We analyze the probability of success and overhead of these techniques (both analytically and by simulations). Our solutions do not substitute cryptography based techniques which generally provide the first line of defense. Instead they compliment the first line of defense. These solutions are necessary because physical capture of a sensor node is easily possible.

9.1 Future Work

We identified following issues as a future work:

9.1.1 Trusting Anomalies and Intrusion Claims

A malicious node can send faulty anomaly and intrusion detection information to the base station. Malicious node can claim that actual legitimate nodes are posing attacks. Verifying validity of the anomaly and intrusion detection claims is a challenge. This problem remains unsolved.

9.1.2 Detection of Physical Capture

Detection of physical capture of a sensor node is difficult to detect. Periodic monitoring of neighbors can be used to detect physical capture. This problem is challenging and we list it as a future work.

9.1.3 Intrusion Response Mechanisms

In this dissertation we just focused on detecting attacks and intrusions. Responding to attacks is an important problem and it remains unsolved for WSN, hence we list it as a future work.

REFERENCES

[1] C. Karlof, D. Wagner, “Secure Routing in Sensor Networks: Attacks and Countermeasures,” in Ad Hoc Networks, volume 1, issues 2--3 (Special Issue on Sensor Network Applications and Protocols), Elsevier, September 2003, pp. 293-315.

[2] S. Shakkottai, R. Srikant, N. Shroff, "Unreliable sensor grids: coverage, connectivity and diameter", INFOCOM 2003, Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, April 2003.

[3] S. Kumar, “Classification and detection of computer intrusions”, PhD thesis, Purdue University, 1995.

[4] J. Anderson, “Computer Security Threat Monitoring and Surveillance”, Technical report, James. P. Anderson Co., Fort Washington, Pennsylvania, 1980.

[5] D. Denning, “An Intrusion Detection Model”, IEEE Transactions on Software Engineering, 13(2):222—232, 1987.

[6] A. Mishra, K. Nadkarni, and A. Patcha, "Intrusion detection in wireless ad hoc networks", IEEE Wireless Communications, vol. 11, no. 1, pp. 48-60, Feb 2004.

[7] Y. Zhang and W. Lee, "Intrusion detection in wireless ad hoc networks," ACM MOBICOM, 2000.

[8] S. Bhargava and D. Agrawal, “Security enhancements in AODV protocol for wireless ad hoc networks”, in Proceedings of Vehicular Technology Conference, 2001.

[9] C. Perkins, "Ad-hoc on-demand distance vector routing," in MILCOM '97 panel on Ad Hoc Networks, Nov. 1997.

[10] S. Marti et al., “Mitigating Routing Misbehavior in Mobile Ad Hoc Networks,” Proc. 6th Annual Int’l. Conf.Mobile Comp. and Net., Boston, MA, pp. 255–65.

[11] S. Bansal and M. Baker, “Observation based cooperation enforcement in ad hoc networks,” Technical Report, 2003.

[12] P. Michiardi and R. Molva, “CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks,” Sixth IFIP conference on security communications, and multimedia (CMS 2002), Portoroz, Slovenia., 2002.

[13] Y. Huang and W. Lee, “A Cooperative Intrusion Detection System for Ad Hoc Networks,” In Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03), Fairfax VA, October 2003.

[14] M. Demirbas, Y. Song, “An RSSI-based Scheme for Sybil Attack Detection in Wireless Sensor Networks,” Proc. of the International Symposium on World of Wireless, Mobile and Multimedia Networks, 2006.

[15] E. Ngai, J. Liu, M. Lyu, “On the Intruder Detection for Sinkhole Attack in Wireless Sensor Networks”, Proceedings of the IEEE International Conference on Communications, 2006.

[16] C. Piro, C. Shields. B. N. Levine, "Detecting the Sybil Attack in Mobile Ad hoc Networks," Proceedings of the Second International Conference on Security and Privacy in Communication Networks, 2006.

[17] P. Albers et al., “Security in Ad Hoc Networks: a General Intrusion Detection Architecture Enhancing Trust Based Approaches,” 1st International. Workshop, Wireless Information Systems, Ciudad Real, Spain, Apr. 2002.

[18] A. B. Smith, “An Examination of an Intrusion Detection Architecture for Wireless Ad Hoc Networks,” 5th National Colloqium for Info. Sys. Sec. Education, May 2001.

[19] O. Kachirski and R. Guha, “Intrusion Detection Using Mobile Agents in Wireless Ad Hoc Networks,” KnowledgeMedia Net., Proceedings of IEEE Workshop., pp. 153–58, July 10–12, 2002.

[20] S. Buchegger and J. Le Boudec, “Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes — Fairness In Dynamic Ad-hoc NeTworks,” Proceedings of 13 IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC), Lausanne, CH, June 2002.

[21] Y. Huang and W. Lee, “A Cooperative Intrusion Detection System for Ad Hoc Networks,” Proc. ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03), Fairfax, VA, Oct. 2003, pp. 135-147.

[22] Y. Zhang and W. Lee, “Intrusion Detection in Wireless Ad Hoc Networks,” Proc. 6th Int’l. Conf. on Mobile Computing and Networks, Boston, MA, Aug. 2000, pp. 275–83.

[23] C. Intanagonwiwat, R. Govindan and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks”, Proc. Mobile Computing and Networking, Boston, MA, Aug. 2000, pp. 56-67.

[24] W. Ye, J. Heidemann and D. Estrin, “An Energy-Efficient MAC Protocol for Wireless Sensor Networks,” Proc. IEEE Infocom, New York, NY, June 2002, pp. 1567-1576.

[25] A. Perrig, R. Szewczyk, V. Wen, D. Cullar and J.D. Tygar, “SPINS: Security protocols for sensor networks,” Proc. MOBICOM, Rome, Italy, July 2001, pp. 189-199.

[26] D. Dean and A. Stubblefield, “Using Client Puzzles to Protect TLS”, In Annual USENIX Security Symposium, Washington, D.C., 2001.

[27] Chang H., Atallah M., "Protecting Software Code by Guards", Lecture Notes In Computer Science; Vol. 2320, pp. 160 – 175, 2001.

[28] W. Warneke and S. Bhave, “Smart Dust Mote Core Architecture,” Project Report, Berkeley Sensor and Actuator Center, Berkeley, CA. Available at: http://bwrc.eecs.berkeley.edu/Classes/CS252/Projects/Reports/warnke.pdf

[29] Z. Chen,. A. Khokhar, "Self Organization and Energy Efficient TDMA MAC Protocol by Wake Up For Wireless Sensor Networks", First Annual IEEE Intl Conf. on Sensor and Ad Hoc Communications and Networks, Santa Clara, CA, October 2004.

[30] B. Krishnamachari, D. Estrin, and S. Wicker, “Impact of Data Aggregation in Wireless Sensor Networks,"Proceedings of the 1st International Workshop on Distributed Event-Based Systems, Vienna, Austria, July 2002.

[31] L. Kleinrock and F. A. Tobagi, “Packet switching in radio channels: Part I“, In IEEE Transactions on Communication, volume 23, pages 1400--1416, 1975.

[32] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, D. Culler,”Exploiting the Capture Effect for Collision Detection and Recovery”, The Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II), May 2005.

[33] S. Avancha, "A Holistic Approach to Secure Sensor Networks," Ph. D. thesis, August 2005.

[34] A. Ephremides, J. Wieselthier, and D. Baker, “A design concept for reliable mobile radio networks with frequency hopping signaling”, Proceedings of the IEEE, 75(1):56-73, Jan. 1987.

[35] http://www.tinyos.net/

[36] Johnson, D. and Maltz, D., "Dynamic source routing in ad hoc wireless networks," in Mobile Computing, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[37] Jourdan, D.B. and de Weck, O.L., “Layout Optimization for a Wireless Sensor Network using a Multi-Objective Genetic Algorithm”, IEEE Semiannual Vehicular Technology Conference, Milan, Italy, May 2004.

[38] MICA2 Mote Datasheet, Crossbow Technology, San Jose, CA. Available at: http://www.xbow.com/Products/ Product_pdf_files/Wireless_pdf/6020-0042-05_A_MICA2.pdf. Last accessed in October 2005.

[39] Miller, M. and Vaidya, N., “A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio”, IEEE Transactions on Mobile Computing, May/June 2005, pp. 228-242.

[40] V. Bhuse, A. Gupta, “Anomaly intrusion detection in Wireless Sensor networks”, Journal of High Speed Networks, Issue 1(15), Jan-Mar 2006.

[41] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar, “SPINS: Security Protocols for Sensor Networks, Wireless Networks,” Mobile Computing and Networking, pp. 189-199, 2001.

[42] B. Przydatek, D. Song and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks", In Proceedings of ACM SenSys, 2003.

[43] D. Shukla, L. Chandran-Wadia and S. Iyer, "Mitigating the Exposed Node Problem in IEEE 802.11 Ad Hoc Networks," Proc. IEEE ICCCN’03, Oct. 2003.

[44] H. Chan, A. Perrig, "Security and Privacy in Sensor Networks," Computer, vol. 36, no. 10, pp. 103-105, Oct., 2003.

[45] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar, “SPINS: Security Protocols for Sensor Networks, Wireless Networks,” Mobile Computing and Networking, pp. 189-199, 2001.

[46] The Network Simulator-ns-2, http://www.isi.edu/nsnam/ns/

[47] E.M. Royer and C-K Toh, “A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks”, IEEE Personal Communications, Apr. 1999.

[48] Bo Yu, Bin Xiao, "Detecting Selective Forwarding Attacks in Wireless Sensor Networks", In the proceedings of Parallel and Distributed Processing Symposium, 2006.

[49] Soumya Banerjee, Crina Grosan, Ajith Abraham and P.K. Mahanti, Intrusion Detection on Sensor Networks Using Emotional Ants, International Journal of Applied Science and Computations, USA, Vol.12, No.3, pp.152-173, 2005.

[50] A. Agah, M. Afrand and S. Das "Prevention of DoS Attack in Sensor Networks using Repeated Game Theory", ICWN 2006.

[51] A. da Silva et al. "Decentralized intrusion detection in wireless sensor networks", Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks, Montreal, Quebec, Canada, Pages: 16 - 23, 2005. P. Techateerawat, A. Jennings, "Energy Efficiency of Intrusion Detection Systems in Wireless Sensor Networks", IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp. 227-230, 2006.

[52] R. Shirey, “Internet Security Glossary”, RFC 2828, http://www.faqs.org/rfcs/rfc2828.html, May 2000.

[53] J. Douceur, "The Sybil Attack", In Proc. of the IPTPS02 Workshop, Cambridge, MA (USA), March

[54] F. R. Schreiber, Sybil, Warner Books, 1973.

[55] A. Wood, J. Stankovic, “A Taxonomy for Denial-of-Service Attacks in Wireless Sensor Networks,” Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, CRC Press, 2004 (invited paper).

[56] V. Bhuse, A. Gupta, “Lightweight Intrusion Detection: A Second Line of Defense for Unguarded Wireless Sensor Networks”, submitted to 2007 IEEE Symposium on Security and Privacy, May 2007.

[57] V. Bhuse, A. Gupta, and A. Al-Fuqaha, “Detection of masquerade attacks for Wireless sensor networks”, accepted for publication at IEEE ICC, June 2007.

[58] V. Bhuse, A. Gupta, and L. Lilien, “Research Challenges in Lightweight Intrusion Detection for Wireless Sensor Networks”, 8th International Symposium on System and Information Security, Nov. 2006.

[59] L. Lilien, Z. Kamal, V. Bhuse and A. Gupta, “Opportunistic Networks: The Concept and Research Challenges in Privacy and Security”, International Workshop on Research Challenes in Security and Privacy for Mobile and Wireless Networks, March 2006.

[60] V. Bhuse, A. Gupta, and L. Lilien, “Detection of packet dropping attack for wireless sensor networks”, the 4th International Trusted Internet Workshop, HiPC, Dec. 2005.

[61] V. Bhuse, A. Gupta, M. Terwilliger, Z. Yang and Z. Kamal, “Using routing data for Information authentication in sensor networks”, with Mark, Ajay Gupta, Zille Huma Kamal and Zijiang Yang, 3rd International Trusted Internet Workshop, HiPC, Dec. 2004.