Attacks and Defenses of Ubiquitous Sensor Networks

S. Shankar Sastry
Tanya Gazelle Roosta

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-58
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-58.html

May 20, 2008

Attacks and Defenses of Ubiquitous Sensor Networks

by

Tanya Gazelle Roosta

B.S. (University of California, Berkeley) 2000
M.S. (University of California, Berkeley) 2004

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Science

and the Designated Emphasis

in

Communication, Computation, and Statistics

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Shankar Sastry, Chair
Professor David Wagner
Professor Suzanne Scotchmer

Spring 2008

The dissertation of Tanya Gazelle Roosta is approved.

Chair Date

Date

Date

University of California, Berkeley

Spring 2008

Attacks and Defenses of Ubiquitous Sensor Networks

by

Tanya Gazelle Roosta

Abstract

Attacks and Defenses of Ubiquitous Sensor Networks

by

Tanya Gazelle Roosta

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Science

and the Designated Emphasis

in

Communication, Computation, and Statistics

University of California, Berkeley

Professor Shankar Sastry, Chair

Based on recent technological advances, the manufacturing of a large number of low cost wireless sensors became technically and economically feasible. Thousands of these sensors can potentially be networked as a wireless sensor network for many applications that require unattended, long-term operations. One of the critical challenges to making sensor networks more pervasive and secure is the severe resource constraints, in terms of energy and memory, on the sensor nodes.

This dissertation explores specific security issues associated with sensor networks. In particular, we explore four related themes: 1) we begin by developing a taxonomy of security attacks and existing countermeasures for sensor networks. Although this taxonomy serves as a reference for security attacks, it points out a lack of a holistic view of the overall *security requirements* and *threat models* in sensor networks. Without these notions we cannot evaluate

the tradeoffs between resource constraints and security. Then, we explore the development of methodologies for evaluation and design of secure sensor network security by defining: (a) security properties and security metrics to help us understand the value of each security solution, (b) a realistic threat model to understand the practical nature of the adversary model in sensor networks, (c) a security design space to identify best practices for the design and configuration of secure sensor networks. This framework can be used to formally define and analyze security attacks and the effectiveness of solutions for each attack and to identify the *path of least resistance* for an attacker. 2) Our second theme explores the issue of insider attacks on fundamental services and applications in sensor networks. This type of attack has a more serious impact on the network since the attacker is in possession of the cryptographic keys and can participate in communication. We specifically look at the time synchronization service and the object tracking algorithm. Time synchronization protocols provide a mechanism for synchronizing the local clocks of the nodes in a sensor network. Many applications, such as networking protocols, rely heavily on accurate timing to perform their tasks. We analyze attacks on different categories of time synchronization protocols, show how these attacks affect different classes of protocols, and propose solutions for each attack. We also implement our attacks and countermeasure for one class of time synchronization protocols. Next, we analyze the effect of insider attack on multiple object tracking by focusing on a hierarchical target tracking algorithm specifically designed for sensor networks. We develop a hierarchical reputation system framework that helps detect node misbehavior and isolate malicious entities. We evaluate our reputation system experimentally and demonstrate how it improves object tracking in the presence of malicious nodes. 3) The third theme in this dissertation deals with the security issues facing the applications that use sensor networks. We look at two important applications that use sensor networks: health care systems, and the process control systems. We develop an integrity monitoring system for the health care application. We develop two security solutions for process control systems: 1) a model-based intrusion detection system, and 2) secure key management and software update. 4) In the last

part of the dissertation, we use a game theoretic framework to analyze and build a distributed reputation mechanism for sensor networks. Game theory provides a way of mathematically formalizing the decision-making process. However, there has been very limited research in the area of sensor network security. Therefore, the object of our research is to analyze the available game theoretic approaches for reputation systems and apply those to field of sensor networks.

_____

Professor Shankar Sastry
Thesis Committee Chair

This thesis is dedicated to my family, without whose love and support I could not have come this far.

Contents

List of Figures

List of Tables

Acknowledgements

I am deeply indebted to my advisor Professor Shakar Sastry whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this dissertation. He is a true inspiration and has been a great advisor and mentor.

I would like to thank my parents, Jaleh and Majid for their continued love, support, and encouragement. I also would like to thank my husband, aunt, and uncle for being there for me when I needed support.

Part 1

Security Paradigm

Chapter 1

Introduction

In this chapter we first discuss what sensor networks are some of their typical application areas. Next, we briefly discuss the most demanding challenges in sensor networks. In particular, these challenges call attention to the importance of ensuring security in sensor networks as a fundamental prerequisite for a functional network and a crucial facet of any practical sensing system. We close the chapter by enumerating the specific research contributions presented in this dissertation.

1.1 Sensor Networks

Due to recent technological advances, the manufacturing of small sensors became technically and economically feasible. Thousands of these sensors can potentially be networked as a Wireless Sensor Network (WSN). A WSN is an ad-hoc, infrastructure-free, multi-hop wireless network where every node can be either a host or a router forwarding packets to other nodes in the network. These networked microsensors provide the technology for a broad spectrum of applications that require unattended, long-term operations. Sensors can be deployed under the water, in the air, under the ground, on the ground, in vehicles, or on the

human body. Some current applications of sensor networks include providing health care for the elderly, surveillance, emergency disaster relief, detection and prevention of chemical or biological threats, gathering battlefield intelligence, and critical infrastructure.

A sensor network's general purpose is to monitor the environment and detect events of interest. Each node is equipped with embedded processing units, a radio, and potentially multiple onboard sensors, such as passive infrared (PIR), magnetometric, seismic, and acoustic. A typical example of a sensor node, sometimes called a mote, is the IMote2, which has a 32-bit processor running at 13 MHz, with 256 kB SRAM, 32 MB flash memory, and 32 MB SDRAM. The radio has an integrated 802.15.4 radio and an external 2.4 GHz antenna [7], as shown in Figure 1.1.

1.1.1 Challenges

The limited energy, bandwidth, and computational resources of these sensors make it difficult to implement various protocols and applications on these networks. The challenge of resource constraints is compounded with the challenge of reliably communicating over an unreliable wireless channel. Sensor networks are usually deployed in environments where the channel is affected by harsh fading. Given the energy, memory, and processing constraints on the motes it is not possible to use sophisticated antennas or modulation schemes. Yet another challenge in sensor networks is to design communication protocols that can scale from tens to thousands of motes. Finally, sensor networks typically are physically unattended after deployment. As a result, the nodes are vulnerable to physical capture and compromise. All of these issues combined make it difficult to design energy/memory efficient, scalable communication protocols which are also secure.

These challenges associated with sensor networks can be summarized as follows:

- **Reliability and cost:** Increasing the density of nodes deployment in sensor networks

Figure 1.1. IMote2.

enables a more accurate monitoring of the environment. However, this will increase the cost of the system deployment and maintenance. There is a need to reduce the cost of system components. On the other hand, the less expensive components are typically less reliable. In many applications of sensor networks, reliability should be above a certain point to assure that the measurements from the deployment environment are corresponding with the real state of the physical system being studied.

- **Prior deployment knowledge:** It is crucial for the nodes in a sensor network to have knowledge of the network topology. Specifically, each node has to know the identity and location of its neighbors, as well as its own location. Given that sensor networks are generally ad-hoc, the location discovery is done either through the global positioning system (GPS) or through the use of localization protocols.

- **Low energy operation:** Deployment of ad-hoc wireless nodes eliminates the need for a fixed infrastructure for sensor networks. However, the wireless nodes need to have a source of energy that lasts sufficiently long time. Techniques for operating in low energy states are mandatory to ensure an efficient usage of the limited energy resources on the sensor nodes. Numerous approaches have been proposed for saving the power

of the nodes by placing them in some type of sleep mode. Another efficient method for saving the energy of nodes is through data compression. On the other hand, the emergence of new technologies for designing energy scavenging devices are promising future devices that may be much more energy efficient.

- **Signal and information processing:** Sensor networks are deployed for monitoring the environment and gathering information about certain events. Given the energy constraint of the nodes, and the high cost of communication, the nodes have to locally process the information they collect. This processing will result in compressed (fused) data which can be sent back to the network operator. Efficient collaborative signal processing algorithms are required to enable this local fusion of data.

- **Security:** Sensor networks are to be used in many sensitive applications that must contain secure data. Examples of such applications include battlefield planning, health care, critical infrastructure and building and transportation security. Furthermore, the ad-hoc and distributed nature of sensor networks introduces new demands on security mechanisms, protocols and systems. There is a need to identify the security gaps in sensor networks and to ensure a secure deployment and operation of the system. Security must be built into the design of the network and not added as an afterthought.

- **Privacy:** In addition to security, integration of sensor networks in critical infrastructure or health care monitoring raises a high level of privacy concerns. The sensor network designers should satisfy the privacy needs of the exposed entities in the system, so as not to hamper the privacy of the people who are involved and use the sensor network or are being monitored by the system.

1.1.2 Main Contributions and Thesis Organization

As mentioned in the Section 1.1.1, the security and privacy of sensor networks is a great challenge. Security is a difficult task to tackle in any network. However, the problem of security in senor networks is compounded due to limited communication and computation power, wireless nature of the communication, and the unattended nature of these networks. In this dissertation we focus on the problem of security in sensor networks. The main contributions of this dissertation are:

1. Security paradigm:

    - Developing a complete[1] taxonomy of security attacks in sensor networks. This taxonomy helps identify future areas of research. In addition, it can serve as a reference manual for anyone interested in deploying these networks.
    - Providing a systematic approach to the categorization and ranking of security attacks, and defining the proper design space based on the security requirements of the application. In order to do the ranking, we look at the specific application of sensor networks in SCADA systems. Our ranking criteria is based on how severe the damage of the attack is on the availability of the network and the integrity of the data collected by the network. The design space is also defined to be proper with respect to this application of the sensor networks.

2. Protocol security

    - Providing a study of time synchronization attacks in sensor networks, and the effect of these attacks on various applications. Various time synchronization protocols have been studied for security flaws and possible countermeasures for each attack have been proposed. In addition, experiments have been run on a real sensor network testbed to verify the extent to which time synchronization attacks

[1] It is important to note that the taxonomy is only 'complete' as of the time of writing of this dissertation. More new attacks can be introduced in the future with a wider deployment of WSNs.

affect the clocks of each sensor node. Finally, we implemented some of the proposed countermeasures for validation.

- Extending some of the existing tracking protocols in sensor networks to include robust procedures to alleviate the problem of insider attacks.

3. Application security

   - Developing a key management protocol for Process Control System (PCS) (also known as SCADA systems). Recently sensor networks have been incorporated into the SCADA systems to decrease the cost and increase data gathering efficiency. However, the security issues that the incorporation of WSN in these systems can introduce have not been studied extensively. In this dissertation, we develops a key management protocol for secure rekeying and a secure software update scheme.

   - Developing a model-based intrusion detection system for the SCADA systems. We define rules and policies for our intrusion detection system. These rules are designed based on the specific characteristics of the sensor networks deployed in an SCADA environment.

   - Developing an integrity monitoring system for sensor network deployed for health care monitoring, such as at-home patient monitoring. The health care application is gaining popularity due to its cost-cutting benefits as well as the autonomy it provides for the elderly patients. Given the sensitivity of the application, in terms of its impact on an individual's health, it is important to design an integrity monitoring system that would monitor the collected data for anomalies. These anomalies can arise from either malicious attacks on the sensors, or from hardware malfunction.

4. Analytical tools for security

- Game theory: We formulate and solve a game of cooperation in the context of routing in sensor networks. Analytical tools have been scares in the area of sensor network security. Game theory offers a rich set of tools for analyzing situations where two players with opposing interests interact with one another. We use the Baysian game theoretic framework to model the scenario where the network of sensors is playing against an intelligent malicious node (i.e. attacker). The solution of this game results in well-defined strategists that the network can use to infer the next move of the malicious node and decide whether to continue its interaction with the malicious actor or not.

The rest of this dissertation is organized as follows. The taxonomy of security attacks in sensor networks is discussed in Chapter 2. The systematic categorization and ranking of attacks, as well as the design space approach to sensor network security are presented in Chapter 3. Chapter 4 discusses the time synchronization attacks in sensor networks, the effect on various applications, and the implementation of attacks and countermeasures on our sensor network testbed. Robust tracking procedures are presented in Chapter 5. Chapters 6 presents the details of our key management protocol and intrusion detection system for PCS. Chaper 7 discuss the integrity monitoring system developed for sensor networks deployed in patient monitoring. Chapter **??** discusses reweighted sum-product algorithm and its uses in sensor networks; then we provide our results on the convergence of the algorithm. Finally, Chapter 8 presents a game theoretic formulation of routing in sensor networks and the solution to the game.

Chapter 2

Taxonomy of Security Attacks in Sensor Networks

The goal of this chapter is to provide a taxonomy of attacks on sensor networks and outline possible solutions for each attack. A comprehensive taxonomy can provide a good first step towards identifying major threats and vulnerabilities in any network, and of interest to us, sensor networks. In addition, this chapter is meant to provide: 1) directions for future research in the area of sensor network security, and 2) a starting point for a systematic categorization of security properties, attacks, and design space in sensor networks[1][2]. It has to be noted that the threat model, trust model, and the security objectives presented in this chapter are very general. Depending on the application at hand, more details can be incorporated in these requirements and models to further improve them. We will elaborate on this point later on in the chapter.

The rest of the chapter is organized as follows. In Section 2.1 we discuss the basic threat model, trust model, and security objectives in sensor networks. In Section 2.2, physical at-

[1]Permission to use parts of these works for the purposes of this dissertation have been granted by the IEEE and/or the appropriate copyright holder(s).

[2]The contents of this chapter are based on [8], and partly on [9].

tacks on the sensor nodes and attacks on TinyOS are covered. Attacks on the communication stack are explained in Section 2.3. Traffic analysis attacks are described in Section 2.4 followed by attack on key management protocols in Section 2.5. Sybil attack is covered in Section 2.6. Attack on reputation schemes are explained in Section 2.7. The attacks on in-network data aggregation are covered in Section 2.8. Finally, we explain the attacks on time synchronization protocols and the effect on the higher level application in Section 2.9.

2.1 Problem Statement

Here we describe the widely used threat and trust model along with the security objectives in sensor networks.

2.1.1 Threat Model

A general way to categorize the attacks in a sensor network is as follows (in [10] some of these categories have been suggested for attacks on routing protocols):

- A ***mote-class*** **attacker vs. a** ***laptop-class*** **attacker**: A mote-class attacker has access to a few motes with the same capabilities as other motes in the network. A laptop-class attacker has access to more powerful devices, such as laptops. This will give the adversary an advantage over the sensor network since it can launch more serious attacks.

- An ***insider*** **attacker vs. an** ***outsider*** **attacker**: An outsider attacker has no special access to the sensor network, such as passive eavesdropping, but an insider attacker has access to the encryption keys or other code used by the network. For example, an insider attacker could be a compromised node which is a legitimate part of the sensor network.

- ***Passive* vs. *active* attacker**: A passive attacker is interested in collecting sensitive data from the sensor network, which compromises the privacy and confidentiality requirement. This information might be used later to launch an active attack. The attacker can eavesdrop packets in transit and analyze them to gather required information. This attack is easier in a wireless environment than a traditional wired network. In contrast, the active attack goal is to disrupt the function of the networks and degrade the performance. Generally speaking, most of the active attacks result in denial of service attack (DoS). Examples include (and more will be discussed later) jamming, hijacking (where an attacker takes control of a communication between two entities and masquerades as one of them), For example, the attacker might inject faulty data into the network by pretending to be a legitimate node. It has to be noted that passive or active attacker mostly refers to the method of attack and is not the goal of an attack. For example, and eavesdropper might be trying to learn some secret about the network whereas the attacker who injects faulty data might be trying to skew the result of some protocol.

Even though this categorization has been suggested for the routing protocols, it could be used in other settings as well. For example, if we are dealing with an aggregation protocol in sensor networks, such as the one in [11], the attacker model could also include injection of faulty sensor readings. However, the above categorization is still applicable since the attacker can either have laptop-class or mote-class capabilities. In addition, in this scenario, the attacker can be an insider and an active attacker. Therefore, we can use the above categorization as a general guideline and first step toward developing a more complete and detailed attacker model that fits the application under consideration.

2.1.2 Trust Model

In sensor networks, there are one or more base stations, such as PCs, which are sinks and aggregation points for the information gathered by the nodes. Base stations are the in-

terface between the sensor network and the users and are often connected to a larger and less resource-constrained network. It is generally assumed that the base stations are trustworthy[3]. Besides the base stations, there are no trust requirements on the sensor nodes since they are vulnerable to physical capture and other attacks.

2.1.3 Security Objectives

Security objectives in sensor networks are very similar to security requirements for embedded systems and are summarized below:

- **Data Confidentiality**: In many applications, sensor networks gather sensitive data, for example in health care or military applications. Data confidentiality means that data is protected and will not leak outside of the sensor network and be used by unauthorized parties. The goal of data confidentiality can be achieved using cryptographic schemes, such as symmetric or asymmetric data encryption methods.

- **Data Authentication**: This requirement allows the receiver to verify that the data was really sent by the node it claims to be coming from. The goal of data authentication can be achieved using a message authentication code (MAC) on the communicated data.

- **Data Integrity**: This ensures that the data has not been altered or modified by unauthorized users while in transit. One method to achieve the goal of data integrity is to use one-way hashing methods [12] before encrypting the data.

- **Data Freshness and Availability**: Given that sensor networks are used to monitor time-sensitive events, it is important to ensure that the data provided by the network is fresh and available at all times. This means that an adversary can not replay old messages in the future.

[3] Although it is comprehensible that a base station could get compromised too.

- **Graceful Degradation**: This might not be considered one of the typical security objectives. However, depending on the sensor network application, it might be important to ensure that the designed mechanisms are resilient to node compromise, and the performance of the networks *degrades gracefully* when a small portion of the nodes are compromised.

2.2 Attacks on Sensor Nodes

Sensor networks are self-organizing networks which, once deployed, are expected to run autonomously and without human attendance. Sensor nodes are vulnerable to physical tampering and capture. The physical tampering in turn facilitates attacks on the sensor software. Each of these attacks are explained in more detail in the following subsections.

2.2.1 Physical Tampering

Current sensor hardware does not provide any resistance to physical tampering due to economic reasons (keeping the cost of sensor hardware down). This lack of tamper resistance provides an attacker with a window of opportunity to capture a mote or a subset of motes. Once the motes are physically compromised, the attacker can easily extract the cryptographic keys as well as exploit the shortcomings of the software implementation.

In the realm of sensor networks, the physical attacks can be divided into two types:

- *Invasive Attacks*: This type of attack consists of reverse engineering followed by probing techniques that require access to the chip level components of the device. As a result, the attacker has an unlimited access to the information stored on the chip, and can cause substantial damage to the system.

- *Non-invasive Attacks*: In this type of attack the embedded device is not opened and

physically tampered with. An example of this type of attack is the side-channel attack. A side channel attack refers to any attack that is based on the information gathered from the physical implementation of a cryptosystem, in contrast to a vulnerability in the algorithms. For example, the attacker may analyze the power consumption, the timing of the software operation execution, or the frequency of the Electro Magnetic (EM) waves.

As of yet, there is no solution available to make sensor nodes resistant to physical tampering; the sensor nodes' micro-controllers lack any kind of memory protection. Traditionally in embedded systems cryptoprocessors, which are physically secure processors, have been extensively used to provide some level of physical tamper resistance. Even though there are known attacks on cryptoprocessors, they do provide a first line of defense against physical tampering. Therefore, there is a need to develop optimized cryptoprocessors that fit the low-cost, low-energy requirements of sensor networks.

Non-invasive attacks, such as side-channel attacks, are also possible in sensor networks. For example, a recent study has shown that a side-channel attack on MAC, using simple Power Analysis as well as Differential Power Analysis, is possible in sensor networks [13]. Their results suggests that several key bits can be extracted through the power analysis attack. This leads to the conclusion that protecting block ciphers against side channel attacks is not adequate. The future research has to explore possible security measures for MAC as well.

Other side-channel attacks which have not been explored in the context of sensor networks are timing attacks and frequency-based attacks. The timing attack involves algorithms which have non-constant execution time and this can potentially leak secret information. Non-constant execution time can be caused by conditional branching and various optimization techniques. As we will outline in the next subsection, the operating system running on the sensor nodes is event-driven and extremely optimized in terms of memory consumption. This suggests that the timing side-channel attack is possible. A solution to this attack is to use

constant execution time software. However, it is not clear if this is easily achievable in sensor networks. Therefore, searching for countermeasures for the timing attack in sensor networks is an important area for future research. In addition to timing attack, frequency-based attacks to extract secret keys of symmetric cryptographic algorithms needs to be further explored in the context of sensor networks. It has been shown in [14] that the frequency-based attack can be carried out on small devices, such as PDA. However, no experiment has been carried out to verify whether this attack is plausible in sensor network context.

Another problem that can arise in sensor network is attacks on the block cipher (in cryptography, block cipher refers to a symmetric key cipher that operates on fixed length groups of bits). TinySec [15], which is one of the encryption mechanism proposed for sensor networks, uses block cipher. Attacks on the block cipher are usually accomplished through linear or differential crypto-analysis. As we mentioned earlier, this attack is possible in sensor networks by using power analysis techniques. In addition, if the block cipher is used as a hash function, attacking the block cipher will result in breaking the hash function.

Given the discussion in this section, it is obvious that there needs to be more research done to prevent these attack. Some of the countermeasures for side-channel attacks used in traditional and embedded systems are:

- power consumption randomization
- randomization of the execution of the instruction set
- randomization of the usage of register memory
- CPU clock randomization
- using fake instructions
- using bit splitting

Future research should look into each of these solutions to determine their applicability to the sensor node platforms. Future sensor node hardware has to be designed so as to support the security required by the software.

2.2.2 Environment Tampering

Another physical attack is environmental tampering. The adversary in principle can compromise the integrity of the sensor readings by tampering with the deployment area. For example, he can place a magnet on top of a magnetometer, or tamper with the temperature of the environment around temperature sensors. This actuation in the sensor network environment may affect the sensor readings registered by the nodes, either accidentally or maliciously. Other types of attack in this category are node displacement and installing new sensors which we will discuss in more detail in later sections.

2.2.3 Software Attacks

Software-based attacks are concerned with modifying the the code, and exploiting known vulnerabilities. A well-known example of this type of attack is the *buffer overflow* attack. Buffer overflow attack refers to the scenario where a process attempts to store data beyond the boundaries of a fixed length buffer. This results in the extra data overwriting the adjacent memory locations.

As mentioned earlier, sensor networks are resource constrained, for example in terms of energy and memory (RAM and flash memory). Consequently, a new OS called TinyOS has been specifically designed for these networks. TinyOS is a low-power, event-driven OS which consists of code that can be reused. TinyOS is a set of components that can be wired together as needed by the application, as shown in Figure 2.1 [16]. The implementation language of TinyOS is NesC, which is a component based language with event-based execution model

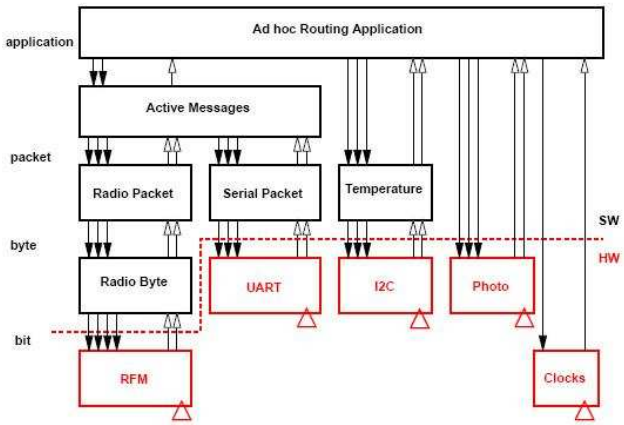


Figure 2.1. TinyOS component hierarchy

[17]. The current implementation of TinyOS does not provide any memory access control, meaning there is no function to control which users/processes access which resources on the system, and what type of execution rights they have. In TinyOS the assumption is largely that a single application or user controls the system.

The solution to the access control in traditional Operating System has been to authenticate the processes , and then mediate their access to different system resources. Another example is to use *Protection Ring* method. A protection ring consists of a number of hardware-enforced levels, for example $0, ..., m$, of privilege within the architecture of a computer CPU. These ring levels are arranged in a hierarchy starting from the most trusted process (usually at level 0) to the least trusted process (usually at level m). The hardware that uses protection ring greatly restricts the ways in which one ring passes control to another ring.

In addition, the hardware enforces restrictions on the types of memory access that can be performed across rings. In order to effectively implement ring architecture, there needs to be a close cooperation between hardware and software. A possible future direction is to design the hardware platform of the sensor nodes such that it supports the ring architecture.

A recent work in [1] uses the concept of drawing a *red line*, which refers to having a boundary between the trusted and un-trusted code. Their solution, called Un-trusted Exten-

sion for TinyOS (UTOS), uses a concept similar to sandboxing. It provides an environment in which un-trusted, and possibly malicious, code could be run without affecting the kernel. UTOS creates the sandbox by using *extensions* which are the interface between the un-trusted code and the TinyOS components. The architecture of UTOS is shown in Figure 2.2. In addition to the above problem, i.e. the non-existence of kernel and user separation, TinyOS uses the concept of Active Messaging (AM). AM is an environment that facilitates message-based communication in distributed computer systems. Each AM message consists of the name of a user-level handler on the target node that needs to be invoked as well as the data that needs to be passed on [18]. This approach enables the implementation of a TCP/IP like network stack on the motes that fits the hardware limitations of the sensor nodes.

Another model for attack is when the adversary opens a port to the nodes and uploads software, or downloads information from the nodes. This vulnerability exists in the current implementation of TinyOS. A port can be opened to a remote sensor node using the USB port and a PC. The *serial forwarder*, which is one of the most fundamental components of TinyOS software, is called to open a port to a node. There is no security check to authenticate the user who is attempting to open the port. This could lead to an attack on the software where

Some of the solutions that should be considered to secure the TinyOS software and protect the software from being exploited by malicious users are:

- Defining rigorous trust boundaries for different components and users
- Software authentication and validation. For example, a new line of research in sensor network has started looking into the problem of *remote software-based attestation* [19].
- Using restricted environment such as Java Virtual Machine. Mate' is already available in TinyOS software [20] and can be used to restrict the access of unauthorized users to the kernel.
- Hardware attestation. For example, Trusted Computing Group and Next Generation

Secure Computing Base provide this type of attestation. A similar model could be used in sensor networks.

- Dynamic run-time encryption decryption for software: this is similar to the encryption/decryption of the data except that the code running on the device is encrypted. This may prevent a malicious user from exploiting the software.

2.3 Attacks on the Network Communication Stack

As explained in [21], the attacks on the communication layer of the network can be divided up into the following categories:

- Physical layer
- Link layer
- Network and Routing layer
- Transport layer

In the following sections, the attacks on each layer of the communication stack is explained in more detail.

2.3.1 Physical Layer

Sensor nodes use Radio Frequency (RF) to communicate through the wireless channel among each other. One of the important attacks on the wireless communication is *jamming*. Jamming is the interference with the RF used by the nodes in a network. The adversary can use a small number of nodes scattered around in the network to disrupt the communication in the entire network. Common defenses against the jamming attack is using some form

of the spread spectrum communication. Examples of this are frequency hopping and code spreading. Another solution to the jamming attack has been proposed in [22]. The authors suggest a mechanism by which a jammed region can be mapped by the surrounding nodes. The goal of the protocol is to cope with the jamming attack, and isolate the jammed region from the rest of the network.

Another attack on the physical layer is eavesdropping. As mentioned earlier, an attack eavesdrops with the hope of collecting useful information from the data packets in transit. The collected information can be used later to launch other attacks, such as denial of service attacks. In [23] the authors classify eavesdropping into two categories:

- **Passive:** the attacker conceals his presence by only listening to the wireless broadcasts among sensor nodes.
- **Active:** the attacker actively tries to query the sensor nodes to gather information form them.

Some general techniques to defend against eavesdropping are: cryptographic methods, and non-cryptographic methods. Cryptographic techniques include encryption and authentication, where encryption protects the contents of the message from being understood by an entity who does not posses the correct keys. Authentication methods, on the other hand, ensures that the message is coming from the node that it claims to be from. Non-cryptographic techniques include injecting fake messages into the network that is indistinguishable from the real messages [23].

2.3.2 Link Layer

Link layer protocol provides means for neighboring nodes to access the shared wireless channel, for example Carrier Sense Multiple Access (CSMA). Examples of attack on the link layer protocol are:

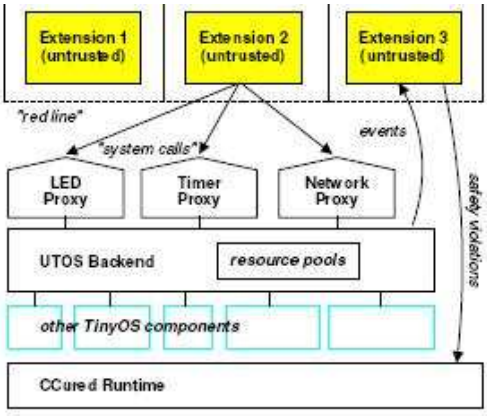


Figure 2.2. UTOS architecture [1]

- Causing collision with packets in transmission
- Exhaustion of the node's battery due to repeated retransmission
- Unfairness in using the wireless channel among neighboring nodes

A number of solutions have been suggested for detecting these attacks, such as using collision detection techniques, modifying the MAC code so as to limit the rate of requests, and using smaller frames for each packet [21].

2.3.3 Network and Routing Layer

The goal of this layer is to provide reliable end-to-end transmission. As mentioned earlier, all the nodes in the sensor network act as routers. This introduces a new complexity dimension to the design of routing protocols for sensor networks. The routing protocols have to be energy and memory efficient but at the same time they have to be robust to security attacks and node failures. There have been many power-efficient routing protocols proposed for sensor networks. However, most of them suffer from different security vulnerabilities as

discussed by authors in [10]. Here we briefly mention a few of the attacks on the routing protocols. The complete list can be found in [10]:

- Black holes: This attack is launched against distance vector routing protocols. A compromised node advertises a zero or a very low cost to its neighbors. As a result, a large number of packets get routed toward this node.

- Wormhole attack: in this attack the adversary node tunnels the messages to another part of the network through a low latency link, and then replays them. This attack is particularly challenging to deal with since the adversary does not need to compromise any nodes and can use laptops or other wireless devices to send the packets on a low latency channel. In [24] the authors propose using *packet leashes*. Packet leashes are additional information added to the packet whose purpose is to restrict the maximum distance the packet can travel in a given amount of time. Another solution has been proposed in [25] where a graph theoretic framework for modeling wormhole links is given. The authors derive necessary and sufficient conditions, based on the graph theoretic framework, for detecting and defending against wormhole attacks.

- Spoofed, altered, replayed packets: This attack targets the routing information used by nodes. As a result, it could lead to creating routing loops, or increase the end to end delay.

- Selective forwarding: in this attack the compromised node only forwards a fraction of the packets it receives and drops the rest. Denial-of-Message attack on broadcast in sensor networks is an example of the selective forwarding.

- Sinkhole attack: in this attack the adversary tries to attract most of the traffic toward the compromised nodes.

- Acknowledgement spoofing: The goal of the adversary in this attack is to spoof a bad

link or a dead node using the link layer acknowledgement for the packets it overhears for those nodes.

2.3.4 Transport Layer

The transport layer is used for managing the end-to-end connections for different applications in the network. Transport layer protocols are usually simplified to fit the requirements of sensor networks, such as energy-efficiency. STCP [26] is an example of a generic transport layer protocol specifically designed for sensor networks.

Flooding and desynchronization are two types of attack targeted at the transport layer protocols. The goal of the flooding attack is to exhaust the memory of a node through sending many connection establishment requests. In the desynchronization attack the adversary forges packets to one or both ends of a connection using different sequence number on the packets. This will cause the end points of the connection to request retransmission of the 'perceived' missed packets. Authentication and using client puzzles are two possible solutions to guard against these attacks [21]. The question that needs to be answered is whether these solutions can be implemented in sensor networks, and what modifications need to be made to make these schemes plausible in the realm of sensor networks.

2.4 Traffic Analysis Attacks

Due to the broadcast nature of the wireless medium, it is relatively easy for an outsider to monitor the traffic in the sensor network and extract useful information regarding the type of the event being monitored, or topology of the network. In this section, we discuss for ways in which an adversary can gain information about the sensor network: 1) traffic pattern, 2) carrier frequency, 3) message size, and 4) message rate [9].

2.4.1 Traffic Pattern

Most sensor networks applications result in a special topology of the network. When an event occurs, the sensor nodes send messages to a base station or a number of base stations. This means that the routing pattern is usually many-to-one or many-to-few. Using this knowledge, an attacker could potentially observe the traffic pattern through the sensor network and infer the location of the base station or other strategically located sensor nodes. In addition, this can reveal information about the location of subjects or objects moving through the sensor network. For example, if shortest path routing is used, the attacker can monitor the sensor nodes in its neighborhood and move from one sensor node to another by following the progression of the forwarded messages. This can be done, for example, by using angle of message arrival. As a result, the location of important nodes in the sensor network are compromised, and the attacker can learn the topology of the network. This in turn results in an attacker learning the physical location of the object being tracked (in the tracking applications of sensor networks). Moreover, due to the multi-hop communication protocols used by many sensor networks, an attacker could trace a stream of messages one hop at a time to get back to the source of information[4]. In many sensor applications, the location-based information can be used to uniquely identify an object or a person, which could result in violation of the privacy of sensor network users. Given that there is a direct correlation between the extent to which an attacker can gain information about the topology of the sensor network and the routing algorithm used, the choice of what routing algorithm to use has to be made wisely.

In short, the traffic analysis attack can be performed in one or both of the following ways:

- In the first attack, the adversary observes the packet-sending rate of the nodes that are close to it, and then moves towards nodes that have a higher packet sending rate.
- In the second attack, the adversary observes the time between consecutive packet-

[4] A sensor mote or motes reporting the location of an object or person.

sendings among neighboring nodes. Then he tries to follow the path of the packet that is being forwarded until it reaches the base station.

Some of the possible solutions to the traffic analysis attack are to use randomness and multiple paths in routing, using probabilistic routing, and the introduction of fake messages in the network. The mechanism for selecting the next hop in probabilistic routing differs from the deterministic routing. In the deterministic routing the next hop is selected based on a known rule, such as shortest path. In contrast, in probabilistic routing, the next hop is chosen at random from among a number of candidate nodes. Therefore, the next hop can not be determined ahead of time. Probabilistic geographic routing scheme (PGR) has been explored by the authors in [27], where the next hop is chosen based on the link quality and residual energy of a subset of the neighbors of a node. It has been shown through simulations that PGR is energy efficient and performs well in terms of throughput.

There have been some routing algorithms suggested in the literature that are specifically designed to mitigate the problem of traffic analysis attack. In [28], the authors use different methods to decorrelate the traffic pattern. One of the solutions proposed is based on the idea of a Random Walk. When a mote receives a message, it forwards the message to one of its parents with probability p, and it uses a random forwarding algorithm with probability $1 - p$. In random forwarding algorithm, the mote selects one of its neighbors with equal probability to forward the message.

A second solution proposed in [28] is to use *Fractal Propagation*. In this method, several fake messages are generated and propagated throughout the network. The objective of having these fake messages is to disguise the true messages. When a mote hears one of its neighbors forwarding a *real* message, it generates a fake message with probability p_f and sends it to one of its neighbors. The combination of Random Walk and Fractal Propagation creates more randomness in the traffic pattern of the network. This reduces the amount of transactional information an adversary can obtain from message routing.

A cautionary note on using fake messages is in order. Fake messages will introduce additional overhead in terms of energy-consumption and in-network traffic. In order for the fake messages to be effective in preventing the adversary from learning any information, they have to look like real messages. Therefore, we can not use any optimization on these fake messages.

One drawback of using these schemes is the amount of overhead they introduce which results in a reduced lifetime of the sensor network. Therefore, designing energy-efficient routing methods that maintain transactional confidentiality in the sensor network are an important future area of research.

2.4.2 Carrier Frequency

Carrier frequency is one area that can be used to compromise transactional confidentiality. It has been shown in [14] that frequency-based side channel attacks can break cryptographic schemes and compromising the content of the message. However, this type of attack can also be used to gather information about the sensor network without needing to know the content of each message in transit. It is conceivable for an adversary that uses a spectrum analyzer to detect the carrier frequency of the data communicated by the motes. This compromises the confidentiality of the network since different sensor platforms use different radio frequencies for communication. For example, the table in Figure 2.3(a), which was taken from the Crossbow Technologies website at: www.xbow.com, shows the carrier frequency of some of the mote platforms.

Once the adversary finds out which carrier frequency the motes are operating at, it can figure out the hardware platform used. Knowing the hardware platform enables the adversary to exploit the software vulnerabilities of the particular version of the software that runs on that mote platform.

In addition, looking at the carrier frequency can help determine what sort of items are

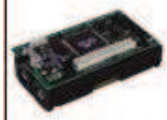
Photo	Commonly Used Name	Frequency Range
	MICA (sometimes refered to as MICA1)	902 to 928 MHz 433.1 to 434.8 MHz
	MICA2	868 to 870; 902 to 928 MHz 433.1 to 434.8 MHz 313.9 to 316.1 MHz
	MICA2DOT	868 to 870; 902 to 928 MHz 433.1 to 434.8 MHz 313.9 to 316.1 MHz
	MICAz	2400 to 2483.5 MHz
	Cricket	433.1 to 434.8 MHz

Figure 2.3. Radio frequency specifications of various sensor hardware platforms.

in operation and who might be using the network. For example, in the United States, the 300-420 MHz band is designated for government use. The 2400-2483.5 MHz bands are used for Industrial, Scientific, and Medical bands(ISM); IEEE 802.11a, 802.11b, and 802.11g Wireless LAN; and IEEE 802.15.4.

The attacker can combine the information from the carrier frequency with additional information, such as routing schemes or message size, to gain knowledge of the sensor network. For example, using the carrier frequency information, an attacker could infer that the sensor network is being used as a government network. Then by looking at the routing scheme, the attacker can determine where the data is being transmitted to.

2.4.3 Message Rate

The rate at which messages are generated by motes in a sensor network can give away information about the nature of the event they are monitoring. Sensor networks can use fixed-rate or event-driven communication protocols. In both cases, an attacker can gain information about the system by observing the message rate.

Since conserving battery power is an important consideration in sensor networks, event-driven protocols are used to conserve the energy of the sensor nodes. Event-driven sensor network applications are such that mote data transmission is triggered by specific events. Using event-driven message generation is one way that information about the context of the sensor network can be leaked. The rate at which these messages are generated can give an adversary specific information about the regularity of certain events being monitored by the sensor network. In specific applications, such as patient health monitoring, this could result in compromise of individual privacy as well.

Sensor networks sometimes use fixed-rate intervals for communication, i.e. the messages are sent periodically. For example, in some tracking schemes, as an object moves from one location to another, the rate at which the motes generate their messages will decrease in one

region and increase in another region. An adversary can observe the rate at which sensor messages are being generated by the motes in its neighborhood, and then move toward the sensor node that has a higher message generation rate. Moving towards the sensor node that has a high message generation rate will lead the adversary to the source, i.e. the sensor node closest to the geographic location of the object [29]. In this case the moving object can be located by the attacker.

When monitoring a sensor node, an aattacker can examine the rate at which it receives messages to determine its distance from the mote. An adversary with knowledge of the message generation rate, for a particular application of sensor networks, can compare this with the rate at which it receives messages. In many cases the message reception rate decreases with increasing distance between the adversary and the mote reporting the information. This allows the adversary to breach network confidentiality. For particular applications of sensor networks, motes are attached to an object that is moving in the network. For example, in the patient monitoring application, bodily sensors are used on a patient's body to monitor vital signs, such as, heart beat or temperature. In many designs within this context, the networked motes send messages at a fixed rate. Therefore, an adversary can determine the distance between this mote and itself and the location of the individual which has the motes attached.

To validate the idea about the message rate, we conducted an experiment using two Crossbow MICAz motes. The MICAz motes have a maximum data rate of 250kbps and a maximum range of about 100 meters. In our experiment, one of the motes acted as a transmitter and another mote was used to intercept the transmitted messages. The motes were placed on top of an empty parking garage that was 150 meters long and 20 meters wide. The transmitting mote was placed at a variable distance away from the intercepting mote. The transmitting mote sent a message every 8 seconds or 7.5 messages per minute. The message reception rate was calculated at the intercepting mote at distances of 10 meters, 30 meters and 50 meters allowing for a period of 5 minutes at each distance. The results of this experiment, as can be seen in Figure 2.4, show that message reception rate falls with increasing distance [9].

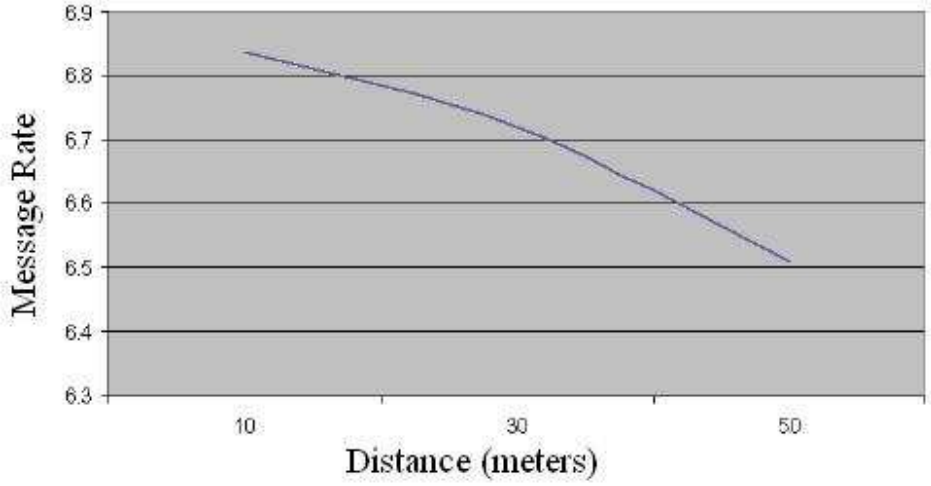


Figure 2.4. The message reception rate versus the distance of the receiver from a transmitting mote.

Therefore, an adversary using a similar intercepting device can determine the radial distance of a mote transmitting at a fixed rate.

2.4.4 Message Size

The size of messages being routed through a sensor network can give information about the network topology as well as what the network is monitoring. For instance, protocols where data gets concatenated or aggregated as it moves from a source mote to a sink mote can provide transactional information for an adversary. A smart adversary can infer multiple things from the changing size of data messages being transmitted through the sensor network. In [30], protocol designs for automatically changing message size based on the amount of data are suggested. In these protocols, the messages are routed with varying sizes in order to decrease energy spent in communication. However, this results in compromise of context information[5] since,

[5]Context information, as opposed to content information, refers the information that can be gathered from the communication characteristics, such as traffic pattern. Content information is what one can learn from the contents of a transmitted message.

- An adversary can infer information about the type of data being communicated and as a result what the network is monitoring.
- An adversary can infer information about the location of a central sink or fusion center

The first item is due to the nature of data being gathered by the network. If the amount of data being collected is small (post-compression), and if the message size is small, due to protocol design, the adversary could infer that the network is collecting slow-varying data (e.g. coarse daily human temperature readings). On the other hand, if data size is large, then message size is large, and the adversary could infer that the network is collecting fast-varying data (e.g. acceleration) [28], [30]. This information can be used by an adversary to determine what is being monitored by the network [28], [30].

The second item is a result of how the data aggregation process works. In-network aggregation of data is an energy saving method in sensor networks [30]. Certain methods of data aggregation could lead to leakage of context information in the sensor network. For example, if the network employs shortest-path aggregation, motes furthest from a fusion center, or sink mote, will rarely participate in aggregating data. However, motes closest to the fusion center or the sink will almost always participate in data aggregation. If message size grows with the amount of data aggregation at a mote, an adversary could gain information about the location of the source motes, fusion centers, or network sinks based on the size of the messages generated around these motes. This could lead to potential security and confidentiality breaches in the network. In particular applications of sensor networks, for instance in tracking, knowing the location of an important mote in the network (e.g. the source mote) could be affected by this security breach.

2.5 Key Management Protocols

Nodes in a sensor network use either predistritbuted keys or use some form of keying material to generate the keys dynamically. The cryptographic keys are sometimes group-wise or pair-wise[6]. In addition, each node has to discover its neighbors with which it shares a secret key. If two nodes do not share a secret key directly, then they have to find a path that connect the two of them in a secure fashion. The goal of the key management protocol is to: 1) equipped the nodes with necessary cryptographic keys prior to the deployment, 2) revoke keys if nodes leave the network, and 3) assign new keys to the nodes joining the network or when some of the keys expire.

One way of categorizing the key management protocols in sensor networks is based on the method used for distributing the keys and other cryptographic material prior to deployment of the sensor nodes:

- Deterministic: In this case the processes that generate the key pools and the key chains are deterministic.
- Probabilistic: The key chains are selected randomly from a given key pool and distributed among the nodes.
- Hybrid: This approach uses a combination of the probabilistic and deterministic solutions to increase resilience and scalability.

In [31], the authors have a different classification of key management protocols:

- Single network-wide key
- Pairwise key establishment
- Trusted base station

[6] Although there could be other options.

- Public key schemes (elliptic curve cryptography)
- Key predistribution schemes, such as random key predistribution schemes

Regardless of the type of the key management protocol and the categorization, all key management protocols must minimally incorporate authenticity, confidentiality, integrity, scalability, and flexibility. Examples of key management protocols can be found in [32], [33], [34], [35]. A comprehensive list of key management techniques and protocols can be found in [31].

The problem with some of the probabilistic approaches such as [33] is that the scheme is not resilient to physical capture attack. Since nodes share pair-wise keys, capturing a small number of nodes is enough to break the protocol. This is due to the fact that in a probabilistic approach, a random subset of nodes are given cryptographic keys prior to deployment. Once deployed, every node tries to build a path from itself to every other node that is composed of secure links, i.e. every node on the path shares a key with the nodes it communicates with. The key pool initially used to predistribute the keys is such that with a very high probability, every node can have a secure path to every other node. However, if a subset of the nodes are physically compromised, depending on the ratio of the compromised nodes to the initial node subset that has the cryptographic keys, the compromised nodes could end up possessing a number of the keys, which results in the compromise of paths among nodes.

Most of the key management protocols are also not resilient to an attacker who listens (eavesdrops) to the communication among nodes during the 'discovery' process of the shared keys. The attacker can use the information he has obtained to attack some of the nodes, and break the key management protocol. This has been shown through simulation in [36]. A possible solution is to use public key cryptography. Recently, a public cryptographic scheme, TinyECC, has been developed and implemented in sensor networks at North Carolina State University [37]. TinyECC is a software package providing Elliptic Curve Cryptography (ECC) operations on TinyOS. In addition, in [38], the authors propose a routing-driven key

management scheme using ECC, which only establishes shared keys for neighbor sensors that communicate with each other. Both centralized and distributed key management schemes are also proposed. Therefore, TinyECC is a promising technique for sensor networks which can help with the problem of key distribution in sensor networks.

2.6 Sybil Attack

Sybil attack refers to the scenario when a malicious node pretends to have multiple identities. For example, the malicious node can claim false identities (*fabricated identities*), or impersonate other legitimate nodes in the network (*stolen identities*). As the authors point out in [39], the Sybil attack can affect a number of different protocols:

- Distributed Storage
- Routing Protocols
- Data Aggregation (used in query protocols)
- Voting (used in many trust schemes)
- Fair Resource Allocation
- Misbehavior Detection

The proposed solutions to Sybil attack include: 1) radio resource testing which relies on the assumption that each physical device has only one radio, 2) random key pre-distribution which associates the identity of the node to the keys assigned to it and validate the keys to see if the node is really who it claims to be, 3) registration of the node identities at a central base station, 4) position verification which makes the assumption that the sensor network topology is static.

Each of these solutions has its own drawbacks. For example, there is no guarantee that every physical device is going to have only one radio. In fact some of the MAC protocols rely on each node having more than one radio. The key pre-distribution is challenging as mentioned in the previous section. The problem with the last proposition is that there is no guarantee that the network topology is static,and the nodes do not change their location. Many sensor network deployment require mobile nodes. Therefore, this solution is likely to fail in the case of dynamic topologies.

2.7 Attacks on Reputation-Assignment Schemes

Reputation or recommendation systems have proven useful as a self-policing mechanism to address the threat of compromised entities. They operate by identifying selfish nodes, and isolating them from the network. They help the users, i.e., the nodes in the network, to decide which nodes they can trust and communicate with. Centralized reputations systems were popularized by the internet. An example of this system is Ebay's rating system.

Decentralized methods were then created for use in ad hoc networks [40]. The *CORE* reputation system [41] and the CONFIDANT protocol [42] use a watchdog module at each node to monitor the forwarding rate of the neighbors of the node. If the node does not forward the message, its reputation is decreases, and this information is propagated throughout the network. Each node also uses the second hand information from other nodes to find the overall reputation of a node. Over time the bad behaving nodes are less trusted and will not be used in forming reliable paths for routing purposes. These two protocols differ in how they use the second hand information, how to punish bad behavior and how to instill trust for the node which misbehave temporarily. A formal model for trust in dynamic networks is introduced by Carbone et. al [43]. The most relevant work for sensor networks is presented in [44]. The authors suggest a high level reputation system frame work for sensor networks. However, they only suggest a *watchdog* mechanism to determine the reputation of each node.

The purpose of the watchdog mechanism is to monitor the neighbors of a node, and determine if any of the nodes deviate from their expected behavior. The authors in [44] state that there is no unified way to design the watchdog mechanism, i.e. the mechanism to assign reputation to each node has to be context dependent and varies based on the application at hand.

Distributed reputation systems face a great challenge since the main assumption behind reputation systems in general is that the information available on an entity of the network is correct. However, there are a number of problems that arise in the reputation-based systems. The following is a short list of the security issues [45]:

- Ballot stuffing
- Bad-mouthing
- Reputation transitivity: if a node can not directly observe another node and assign a reputation, should it trust its neighbors and take their reputation for the unobserved node into account, and how should it combine this second-hand information with its own first-hand information.
- The Sybil attack
- Measuring the performance of the network: how should one assign a reputation value to a node so as to get the desired outcome.

In addition to these difficulties, there are other problems that can arise within the reputation system framework. For example, as mentioned earlier, in [42] the authors propose using a watchdog mechanism that determines if nodes are forwarding their packets properly. Since the medium of transmission is wireless, all the nodes in the neighborhood of a transmitting node can hear the communication. This *overhearing* property is used by the watchdog mechanism to determine misbehavior. However, in a number of other applications in sensor network, we can not utilize the overhearing property. For example, if the network

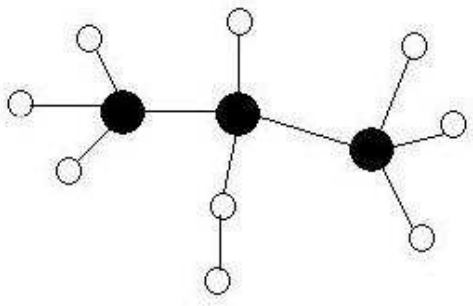


Figure 2.5. The white nodes are sensor nodes, and the black circles are the aggregate nodes.

is used to monitor an event, such as a moving object, then the watchdog mechanism has to be designed in a way to take into account the physical correlation among neighboring nodes' signal strengths (observations) to detect misbehavior. Currently, there are no available analytical models for signal strength correlation based on distance in sensor networks.

Given the security problems facing reputation systems along with the lack of analytical models for designing watchdog mechanisms, the design of a reliable reputation system for sensor networks has proven to be a very challenging problem. Future research needs to focus on designing reputation-assignment mechanisms that are resilient to most of the above security attacks. At the same time these mechanisms have to be simple enough to be implemented on the limited memory of a sensor node.

2.8 Attacks on In-Network Processing

In-network processing, also called data aggregation, is a key feature of sensor networks. Given the limited resources of the sensor nodes, it is nearly impossible for all the nodes to send back their data to the base station. In addition, the rate of packet collision will significantly increase if all the nodes report their observed data. Therefore, some of the intermediate nodes fuse the data from their neighborhood, and send the aggregated data back

to the base station. It is possible to have multiple levels of hierarchy among nodes, and in each level one node fuses the data from the nodes at the level directly below it. An example of the aggregation process is shown in Figure 2.5.

As one can immediately see, if a few nodes are compromised, they can inject faulty data into the network. This will result in a corrupted aggregate. For example, assume that the nodes are monitoring the temperature of the environment, and the fusion process is simply to take the average of all the sensors' readings. Now if one node is compromised and gives a high or a low reading, it will affect the average and pull in one direction. Other, more fundamental, applications that use data aggregation are multi-object tracking [5] and *directed diffusion* routing [46]. Here we will give an example of how a hierarchical multi-object tracking can be affected by attacks on the aggregation. In hierarchical tracking, there are hierarchies of nodes. At each level, a leader is selected and the nodes in the its neighborhood send their observations to the leader. The leader uses an averaging scheme to fuse the observations. The leader nodes, then, send their fused observations to the base station. Base station forms the object's track bases on the received aggregated data. However, if a fraction of the nodes are compromised and send faulty observations to the leader nodes, the fused observation will be skewed. As a result, the track formed by the base station is either wrong, or in some cases non-existent in reality.

There have been a few solutions suggested in the literature to secure the in-network processing [47]–[49]. Wagner [48] suggests using statistical properties, such as median, to reduce the effect of attacks on the aggregation process. Authors in [47] propose a solution based on forming secure hierarchies of node clusters. Cryptographic keys are used at each level of the hierarchy to establish secure communication among the nodes in a cluster. The solution proposed in [49] also relies on cryptographic key establishment to ensure security of the fused data. One possibility to secure in-network data aggregation is to use a reputation system. A node's data is only considered if the reputation is high enough and discarded otherwise. However, using this solution requires having a robust and attack-resistant reputation

system. A second possibility is to use robust statistical methods for estimation that are not as prone to errors as averaging.

2.9 Attacks on Time Synchronization Protocols

Time synchronization protocols provide a mechanism for synchronizing the local clocks of the nodes in a sensor network. There are several time synchronization protocols for the internet, such as Network Time Protocol (NTP). However, given the non-determinism in transmissions in sensor networks, NTP cannot be directly used in wireless sensor networks. Time synchronization implementations have been developed specifically for sensor networks. Three of the most prominent are Reference Broadcast Synchronization (RBS) [50] Timing-sync Protocol for Sensor Networks (TPSN) [51], and Flooding Time Synchronization Protocol (FTSP) [52]. However, none of these protocols were designed with security as one their goals. An adversary can easily attack any of these time synchronization protocols by physically capturing a fraction of the nodes and have them inject faulty time synchronization message updates. In effect, the nodes in the entire network will be out-of-sync with each other.

Time-synchronization attacks have great effects on a set of sensor network applications and services since they heavily rely on accurate time synchronization to perform their respective functions. We will discuss the time synchronization attacks and some of the possible countermeasures in great detail in Chapter 4.

2.10 Discussion

Sensor networks are a promising technology with many important applications, such as environment monitoring, health care, surveillance. They are the way of the future, and it is

envisioned that the senor networks will be used in critical infrastructure. The sensor motes have limited communication and computation resources. The reason for these constraints is that the driving force behind sensor network's success is the small dimension, which facilitates non-intrusive deployment, and the cheap price of the hardware.

The protocols designed for sensor networks have to be simple and efficient both memory-wise and energy-wise to accommodate the resource constraints of the sensor nodes. However, given the unattended nature of sensor networks, they are vulnerable to a number of security attacks which could substantially degrade the performance of the network. The goal of this chapter is to give a comprehensive taxonomy of the security attacks on sensor networks and their effect on the performance of the network. The taxonomy can be used as a reference manual for anyone interested in deploying a sensor network.

For each set of possible security attacks, we outlined possible solutions that have been suggested in the literature or should be considered. In addition, we discussed possible future directions for extending research in the area of sensor network security.

Chapter 3

Systematic Approach to Sensor Network Security

As we saw in Chapter 2, there are numerous possibilities for attacks in sensor networks. If we implement the countermeasures suggested for each of the proposed attacks, the security overhead will overwhelm the available resources of the sensor network, and given the memory and power constraints of the sensor motes, this will not offer a helpful solution. As a result, it seems that attempting to create a *secure* sensor network appears to be an impossible task[1].

This is not a problem unique to sensor networks, but the problem is that deployments of sensor networks have been used mainly for either: (1) scientific purposes, where an adversary has little incentive to attack the sensors, or (2) military deployments, where very little public data is available. As a result, most of the academic research for the security of sensor networks has been done in abstract scenarios, where many assumptions are made and some of which might not hold true in a real world application. For example, the type of threats

[1]The contents of this chapter are based on [53].

the sensor network is exposed to and the architecture and resource constraints of the sensor networks are some of these assumptions.

However, recently sensor networks have found their way into real commercial applications, such as, the process control systems or patient health monitoring, both of which will be covered in later chapters in this dissertation. This offers us the opportunity to use concrete practical scenarios and avoid making assumptions about abstract scenarios.

In this chapter we begin to address the problem of designing a systematic approach to the sensor network security. The contributions of this chapter are:

1. Providing the background setting for the security of sensor networks in Supervisory Control and Data Acquisition (SCADA) systems. We identify (1) the common architecture and resource constraints of the sensor networks, and (2) the incentives an attacker has to attack, and the methods he can use to perform his attacks.

2. Providing a *holistic* view of the *security requirements* and *threat models* of the sensor networks. We express our holistic view with two considerations: (1) we focus on *high-level* security goals (we argue that previous research has focused on *low-level* security goals), and (2) we introduce a class of physical attacks. As seen from Chapter 2, most of the previous research has focused on cyber-attacks.

3. Providing a ranking of threats and security mechanisms. While our rankings may not be general enough, we believe our research is an important first step to better understand the threats against a sensor network and to understand our priorities for protecting them.

4. Defining the *high-level* security goals of a sensor network. While terms like *availability* and *integrity* tend to be understood informally, we provide a precise definition and interpretation of these properties in sensor networks.

5. Identifying different ways that sensor measurements are reported back to the base sta-

tion. We show how event-based sensor measurements can compromise confidentiality of the network.

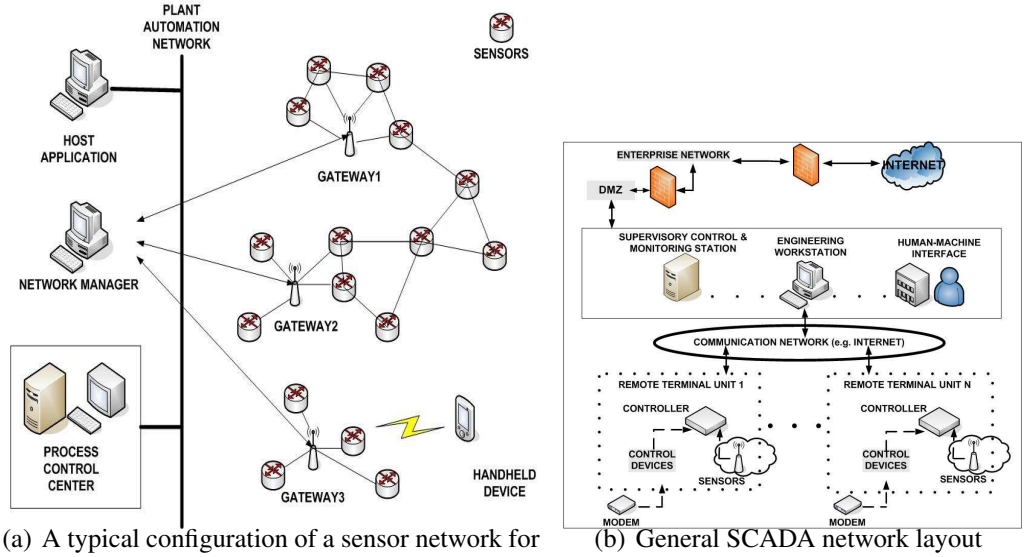
The rest of the chapter is organized as follows: In Section 3.1 we discuss the use of sensor networks in SCADA systems and emphasize the importance of securing sensor networks. Section 3.2 outlines the *security properties* of the sensor network as seen from the point of view of a network user. Section 3.3 describes the *threat model*. The goal is to provide a general framework to analyze the threats, risks and attacks against the global security requirements by determining the conditions necessary for an attack to succeed and its estimated consequences. This framework gives us a way to identify and evaluate the things that can go wrong in the network. In Section 3.4 we study various *security design spaces* to identify best practices for the design and configuration of secure sensor network. Our aim is to help a system designer decide how to best defend the deployed sensor network.

3.1 A Motivating Example: Supervisory Control and Data Acquisition Systems

One of our main motivations is to understand the practical impact of security as sensor networks start transitioning from idealized concepts to concrete practical applications. In this section, we present one example of a commercial application of sensor networks.

Supervisory Control and Data Acquisition Systems (SCADA) refers to large scale, distributed measurement (and control) networks. They are used to monitor or to control chemical or transport processes, municipal water supply systems, electric power generation, transmission and distribution, gas and oil pipelines, and other distributed processes.

A major drawback of typical SCADA systems is the cost of wiring devices together. Wireless sensor networking is a promising technology that can considerably improve the sensing



(a) A typical configuration of a sensor network for SCADA

(b) General SCADA network layout

Figure 3.1. SCADA network

capability of the SCADA system and significantly reduce the wiring costs. Motivated by these incentives, a number of companies have teamed up to bring sensor networks in the field of process control systems, and currently, there are two working groups to standardize their communications [54], [55]. Figure 3.1(a) shows the integration of wireless sensor networks with SCADA systems, and Figure 3.1(b) shows a typical SCADA layout.

The sensor networks used in SCADA systems have a number of general characteristics that are different than the generic characteristics of sensor networks:

- There is an online trusted third party (monitoring station).
- There is no aggregation: the controller collects all the data coming in from the sensors.
- The battery life of the sensors is expected to last several years; therefore, the energy efficiency of the protocols is not as critical as in other applications of sensor networks.
- Although there are some implementations of multi-hop routing, the majority of current deployments use a single hop between the sensors and the gateway.

- Sensors must be accessible and configurable by handheld devices used by network operators.

While the deployment of sensor networks is beneficial for the operation of an industrial control system, deploying wireless devices without security can be dangerous. For example, and adversary may be able to send spoofed packets to a controller, causing it to give undesired outcomes. Because many SCADA systems perform vital functions in national critical infrastructures, such as electric power distribution, oil and gas refineries, and water treatment and distribution, the disruption of control systems could have a significant impact on public health, safety, and lead to large economic losses. Securing control systems in critical infrastructures is thus a national priority [56], [57]. The main security requirements for the SCADA systems are, in the order of importance, availability, integrity and confidentiality. We explore this interpretation in the next section.

The security of SCADA systems will be used as our baseline motivation throughout this chapter. Although our framework can be used for the security of general sensor networks, we use the SCADA system as an example to justify some of our assumptions. This can facilitates the comparison of our assumption and traditional assumptions made in the sensor network literature.

3.2 Security Requirements

The goal of this section is to analyze global requirements, such as, *confidentiality*, *availability*, *integrity*, and *privacy* of the network, instead of focusing only on the requirements for secure middleware (e.g., secure routing) as done in previous research. We classify the goals of a sensor network into two classes: (1) gathering information from a set of sensors in different locations, and (2) preventing the use of the resources of a sensor network by an unauthorized party.

Availability and **integrity** represent the goals of using a sensor network. Availability refers to the ability to collect data at all times that the network is deployed and under operation. Integrity refers to the confidence level of the network operator in the fact that the data collected is correct and has not been altered while in transit.

Confidentiality and **privacy** represent the protection against the possible side-effects of using a sensor network. As we discussed in detail in Chapter 2, an adversary can use the data collected by the network to obtain information that should be secret. An example of this breach of confidentiality due to side information is traffic analysis. This is what was referred to as the context privacy.

Motivated by RFC 2828 [58], we define the security goals in a sensor network more precisely.

1. We define **Service Integrity** as the trustworthiness of the information provided by the sensor network. The quality of the received information allows the sensor network to perform its intended function, which is the collection of accurate data from the sensors. This collection of data is the main *application-layer* service. A violation of service integrity results in **deception**. Deception refers to a circumstance where an authorized entity receives false information about the phenomenon being monitored, and it believes the faulty information to be true.

   Service integrity depends on three factors:

   - **Measurement integrity:** prevents the modification of the sensor measurements. An attack against measurement integrity succeeds when sensor nodes report data that is not representative of their intended environment. False readings can be sent by: (1) a compromised node, (2) affecting the environment around the sensor by the adversary, such as, placing a magnet on top of a magnetometer, or (3) changing the location of the sensor node by the attacker.

- **Message integrity:** prevents unauthorized modification of the data sent from the sensor node, i.e., message *tampering*. Contrary to the measurement integrity, which includes physical attacks, message integrity tries to prevent digital tampering of the messages sent by the node. Besides preventing tampering, message integrity should also provide: (1) **Data liveness:** and (2) **Source authentication:**. Data liveliness is to prevent **replay** attacks by giving the recipient of the data an indication of *when* the sensor reading was measured. Data liveness is usually divided into **weak liveness**, which is when the receiver knows the *ordering* of the sensor readings, but not the exact time), and **strong liveness**. Weak liveness is usually achieved by adding a *message counter* that is incremented when each message is transmitted between a sender and a receiver. Strong liveness can be achieved by polling a sensor with a *nonce*, or by adding a *time stamp*. However, one must be aware of time synchronization attacks [59], [60] and take precautionary measures against it. Source authentication is to prevent **spoofing** attacks by providing the recipient of the data with the evidence of the identity of the message's source.
- **Processing integrity:** This refers to how the gathered data by the sensor nodes is fused before being sent to the main base station or base stations. It is important that the aggregated data is processed in a way that the integrity of the data stays intact. For example, if the sensor nodes are gathering temperature data, and a subset of neighboring nodes average their temperature readings before sending it back to the base station, then the average has to represent the true average of all the sensor readings and not be corrupted.

2. **Network Availability**: the information collected by the sensors is accessible and useable upon demand by a legitimate user. A violation of network availability results in **denial of service**: the prevention of authorized access to the sensor measurements. To understand why industry practitioners consider that the availability of a sensor network

is more important than its integrity, we define **Service Availability** as the union of **Network Availability** and **Service Integrity** –that is, *service availability* ensures that the measurements received are correct. *service availability* should be the top priority of a sensor network, because we need to receive the 'correct' data. Just receiving data, if it is not correct, is not useful for the network administrator. Because most of the literature in sensor networks uses the term *availability* to refer to *network availability*, in the remaining of this chapter we do the same.

3. **Confidentiality**: the information collected by sensor networks is only accessible to authorized users and prevent unauthorized users from learning the information collected by the sensors. **Disclosure** results in the violation of confidentiality. A typical way to ensure confidentiality is by using encryption. However, due to the low-entropy of certain sensor measurement reports, confidentiality may be compromised by simple traffic analysis, as explained in Chapter 2.

4. **Privacy**: the prevention of unauthorized users from learning sensitive *personal* information by using the sensor network. Privacy can be considered a special case of confidentiality when the data collected is personal (context privacy). Examples include, surveillance camera networks, or sensors monitoring the vital signs of patients.

5. **Access Control**: the prevention of unauthorized access to the network. It prevents outsiders (unauthenticated principals) from joining the network, while imposing and enforcing proper restrictions on what insiders (authenticated principals) can do. A security violation of access controls results in **usurpation**: the use of system services or functions by an unauthorized entity.

Confidentiality, privacy, and integrity depend to a large extent on enforcing **access control**. Access control is, however, more general: its goal is to protect *all* the resources of the network, including the misuse of the communication infrastructure. Consider for example

the deployment of two sensor networks A and B within the same wireless range. Access control in network A would prevent the user of network B from using network A to route its packets. In this sense, access control will prevent free riders.

Although we consider *privacy* and *access control* implicitly, the main focus of our work in this chapter is to study service integrity, availability and confidentiality.

3.2.1 High-Level vs. Low-Level Security Goals

All the security properties we have defined represent the general security requirements of the end-user of a network. These can be considered as *high-level* security goals: the goals of using a sensor network in the first place, without looking at the supporting services.

Other security goals, such as, secure routing, secure key distribution, and secure neighbor discovery, can be considered as *low-level* security goals. We argue that most of the research for the security of sensor networks has focused on the design of low-level mechanisms, such as, secure routing protocols, to achieve these low-level security goals. However, there has been very little research effort to try to understand how these low-level mechanisms relate to the high-level security goals.

In this chapter, we attempt to understand this relationship by considering the ranking of the most essential security mechanisms for achieving the high-level security goals. For example, instead of designing a new secure routing algorithm, which is a mechanism intended to provide availability, we ask how much do we gain in *availability* by using a given routing protocol. To answer the question of how to select the appropriate security mechanisms requires a *threat model* of the network. This is because an attacker with the ability to jam the network will achieve a denial-of-service attack no matter which routing protocol is used.

3.3 Threat Model

Achieving perfect security is impossible since an all powerful adversary defeat any security mechanism. In addition, defending against and responding to every possible attack vector will prove to be prohibitively costly. Therefore, it is important to define what we are secure against and what the threat model is.

The goal of defining a threat model is to formalize the perceived **risk** against the network. In literature, risk is defined as the estimation of the likelihood of an attack and the consequences of the attack. The threat model should describe the capabilities of an adversary and identify the threats and attacks against the intended security requirements.

The types of threats in sensor networks may be different than the threats against traditional computer networks. For example, the Internet infrastructure, such as routers and DNS servers, is in general well protected. Functionally important computers are kept in physically secure areas, and there is a level of redundancy and diversity that allows the infrastructure to survive several attacks. In contrast, in sensor networks, the *infrastructure* is the sensor nodes themselves, and they are less protected than traditional infrastructure servers due to physical vulnerabilities discussed earlier.

This perceived vulnerability of sensor nodes has been explored extensively in the research literature. This has lead to the suggestion of a wide range of attacks, which was discussed in Chapter 2. We build upon this previous work by studying the question of: *which are the most likely attacks that an adversary will follow to compromise our security properties*. A very common premise about the *knowledge* of the adversary is to assume it knows all the details of the protocols we use. This assumption is made by popular adversary models, such as the Dolev-Yao model [61], or Byzantine attackers [62]. For our purposes in this chapter, we assume the adversary knows the topology, the identities of the nodes, and the types of sensors we have. The only information the adversary does not have is the secret communication keys.

3.3.1 Threat Taxonomy

In contrast to the classification presented in Chapter 2, here we categorize the attacks into three main types, based on what the end-goal of the attacker is:

- **Outsider attacks**: this type of attack does not require any knowledge of secret keys being used by the network.
- **Key-compromise attacks**: these attacks help the attacker change type, i.e., go from an outside attacker to an inside attacker, by compromising the secret keys used in the network.
- **Insider attacks**: these attacks are the most complex type of attacks given the adversary requires having access to the secret cryptographic keys.

In addition, each attack can be put into one of the two classes: 1) *final* attack, 2) *intermediate* attack. Final attacks are the ones in which the attacker accomplishes its final goal, meaning compromising of one or more of the high-level security requirements. Intermediate attacks, on the other hand, are attacks which facilitate carrying out other attacks. They are used by the adversary as a stepping stone to accomplish his final goal and increase its capabilities.

In order to assess the damage caused by each attack and identify the path of least resistance for the attacker, we categorize each known attack (Chapter 2) in sensor networks into one of the three types. Then, we use a threat ranking scheme to score each attack based on the difficulty of being performed. Finally, we rank the impact of each attack on the main security requirements, i.e. integrity, availability, and confidentiality.

3.3.2 Outsider Attacks

In this section, we categorize some of the typical attacks considered in the literature in terms of whether they are intermediate or final attacks.

Spoofing attack: In this attack, a system entity illegitimately assumes the identity of an authorized system entity. **Attack class**: final.

Jamming attack: Jamming is the interference with the Radio Frequency (RF) used by the nodes in a network. It makes use of the broadcast nature of the communication medium. **Attack class**: final.

Replay attack: In the reply attack, the transmitted packet is maliciously or fraudulently repeated or delayed by the adversary. **Attack class**: intermediate.

Wormhole attack: in this attack the adversary uses his nodes to tunnel the messages to another part of the network through a low latency link. The attacker can use laptops or other wireless devices to send the packets on a low latency channel. Given wormhole is an intermediate attack, and we are assuming that the attacker has no secret keys, its impact depends on the final attack that is carried out by using wormhole. However, this attack could potentially have a higher impact if the adversary is able to infer/distinguish the types of the packets in transmission. By knowing the type of the packets, such as, data, acknowledgement, time update, or advertisement, the adversary can tunnel the 'control' packets and cause more damage to the underlying protocols. It is possible for an attacker to infer the type of the packet in transmission by gathering information as a consequence of generation, transmission, and routing of data messages within the network, such as performing traffic analysis. He can use the message generation rate, message size, and other peripheral information available to him through the broadcast medium to make the inference. **Attack class**: intermediate.

In addition, we introduce the following four *outside-physical* attacks.

Destroying a node: if the sensor network lacks physical access security, it is relatively easy for an attacker to walk up to a node and destroy it. This is an effective attack against availability. The main drawback of this attack from the adversary's point of view is the risk of apprehension. **Attack class**: final.

Environment tampering: The adversary in principle can compromise the integrity of the sensor readings by tampering with the physical environment around sensors. For example, he can place a magnet on top of a magnetometer, or temper with the temperature of the environment around temperature sensors. This is an effective attack against service integrity. The main drawback of this attack is the high risk of apprehension if the network is under some kind of surveillance. **Attack class**: final.

Node displacement: The attacker can change the location of the sensor nodes. By placing the sensor in an incorrect location, the measurements it is going to report to the base station will be erroneous. Therefore, this is an effective attack against service integrity. **Attack class**: final.

Install new sensors: Again, if the area where the sensor nodes are deployed is left unattended, the adversary may be able to install its own sensors and monitor the physical event that we monitor. This is an effective attack against confidentiality. **Attack class:** final.

3.3.3 Key-Compromise Attacks

In this section, we discuss the attacks which enable an adversary to go from an outside attacker to an inside attacker. These attacks are intermediate attacks for availability and service integrity, and final attacks for confidentiality.

Cryptanalysis: This attack refers to transforming encrypted data into plaintext without having prior knowledge of the encryption parameters or processes. In the worst case, the

attacker will obtain the secret keys of a set of devices, and is able to impersonate them.

Exploit: An exploit takes advantage of a software vulnerability to compromise a system. It can be argued that the number of vulnerabilities in sensor networks should be smaller than computers typically connected to the Internet because sensors provide less services. With less functionality and less complex code there will be fewer software bugs. Additionally, because of the resource constraints in sensor networks, programmers have to spend more time per line of code.

Physical tampering: If an attacker has the necessary technical skills and equipment, he could physically compromise the sensor nodes and download the data and other keying material. Additionally, an attacker can succeed in performing a side-channel attack to analyze the physical activities of the system to extract the cryptographic keys.

3.3.4 Insider Attacks

In this section, we describe the insider attacks which require having access to the secret keys used in the network, and categorize them in terms of intermediate or final attacks.

Sybil: Sybil attack refers to the scenario where a malicious node pretends to have multiple identities. For example, the malicious node can claim false identities (*fabricated identities*), or impersonate other legitimate nodes in the network (*stolen identities*) [39], [63]. **Attack class**: intermediate.

Replication: In this attack, the adversary attempts to add one or more nodes to the network that use the same ID as another node in the network [64]. **Attack class**: intermediate.

Denial of service at the link layer or MAC layer: Examples of attack on the link layer protocol are: 1) causing collision with packets in transmission, 2) exhaustion of the node's

battery due to repeated retransmission, 3) unfairness in using the wireless channel among neighboring nodes [65]. **Attack class**: final.

Routing attacks: In these type of attacks, an attacker tries to create routing loops or advertise false routes. The final objective is to degrade the availability of the system, or to receive more traffic for cryptanalysis [66]. **Attack class**: final.

Time Synchronization attack: time synchronization protocols provide a mechanism for synchronizing the local clocks of the nodes in a sensor network. As a result, when there is an attack on these protocols, a fraction of the nodes in the entire network will be out-of-sync with each other [60]. This in turn affects the sensor network applications that rely on tight synchronization to perform correctly, such as, TDMA-based protocols or object tracking [59]. **Attack class**: final.

Slander attack: This attack is only possible if a distributed detection system is implemented, and the sensor nodes can accuse each other of misbehavior. Slander attacks are very dangerous to distributed node revocation techniques [67]. **Attack class**: depends on the system. It will not affect our high-level security properties unless we use a reputation scheme.

Wormhole: 1) in-band wormhole attack, 2) out-of-band. An insider can use a wormhole more effectively than an outsider since it can certainly identify different types of packets. A distributed mechanism for detecting wormhole attack in sensor networks is given in [64]. **Attack class**: intermediate.

3.3.5 Security Metrics

Once the threat model and threat ranking are written, it is important to develop an objective function that captures the overall effect of various attacks on the security requirements, i.e. integrity, availability, and confidentiality. In this section, we define a function that con-

Table 3.1. This table shows whether an attack requires technical knowledge of the sensor network protocols to exploit the vulnerabilities. EXP refers to technical expertise/skills.

Attacks	***EXP Req.***	***No EXP Req.***
Spoofing		✓
Jamming		✓
Destroying the node		✓
Environment tampering		✓
Node displacement		✓
DoS at MAC layer	✓	
Routing	✓	
Time synchronization	✓	
Slander		✓
Sybil	✓	
Cryptanalysis	✓	
Remote compromise	✓	
Replay		✓
Wormhole	✓	

Table 3.2. This table shows whether an attack needs special hardware or can use commodity hardware, such as laptops, computers, or motes. HW refers to hardware in this table.

Attacks	***HW Req.***	***No HW Req.***
Spoofing		✓
Jamming	✓	
Destroying the node		✓
Environment tempering	✓	
Node displacement		✓
DoS at MAC layer		✓
Routing		✓
Time synchronization		✓
Slander		✓
Sybil		✓
Cryptanalysis	✓	
Remote compromise	✓	
Replay	✓	
Wormhole		✓

Table 3.3. This table compares which attacks need physical access and which ones do not. It has to be noted that although some attacks do not explicitly require physical access, the attacker should have gained access to the sensor nodes prior to launching these attacks. We denote these by prior physical access in the table. PA refers to physical access.

Attacks	***PA Req.***	***No PA Req.***	***Prior PA***
Spoofing		✓	
Jamming		✓	
Destroying the node	✓		
Environment tempering	✓		
Node displacement	✓		
DoS at MAC layer			✓
Routing			✓
Time synchronization		✓	✓
Slander			✓
Sybil			✓
Cryptanalysis	✓		
Remote compromise		✓	
Replay	✓		
Wormhole		✓	✓

sists of a weighted combination of three factors: impact on integrity, impact on availability, and impact on confidentiality.

$$
\begin{aligned}
& f(\alpha, \beta, \gamma) = \alpha \times a_1(.) + \beta \times a_2(.) + \gamma \times a_3(.) \\
& a_1(x, x') = \textstyle\sum_i \int \parallel x_i(t) - x_i'(t) \parallel^2 dt \\
& a_2(n_2, t) = n_2 \times T \\
& a_3(n_3, s) = n_3 \times s
\end{aligned}
\tag{3.1}
$$

where

- $a_1(.)$ measures the amount of compromise in integrity. $x_i(t)$ is the true value of the physical process we are monitoring by node i at time t, and $x_i'(t)$ is the value the attacker manages to send to the base station.

- $a_2(.)$ measures the amount of compromise in availability, which is a function of how many packets per second are intercepted and how long the attack continues T.

- $a_3(.)$ measures the amount of compromise in confidentiality. The arguments to the function are the number of nodes compromised (n_3) and the sensitivity of the data (s) which is application-dependent.

Furthermore, (α, β, γ) is a vector composed of weights of the security objectives. For example, in our SCADA system scenario, we are more interested on compromising integrity and availability. In the remaining of this section we analyze the security metrics and the difficulty of launching attacks against integrity and availability. A similar analysis can be performed for confidentiality.

We give an example of how to use this objective to formalize the damage caused by a jamming attack on the SCADA sensor network. Jamming is an attack on availability as opposed to integrity or confidentiality. Therefore, the objective function becomes $f(\beta) = \beta \times n_2 \times t$. α and γ are equal to zero for and β can be set to one to simplify the formula. The number of nodes affected by the jamming attack are proportional to the area covered by the jamming device, which depends on how powerful the radio of the device is. We assume that the jamming radius is R, and the sensor deployment density is $\rho = \frac{\text{\# of nodes}}{\text{deployment area}}$. Using these values, the number of nodes affected by the jamming device are $n_2 = \rho \times \pi R^2$. If the jamming is continued for t seconds, and the average number of packets in transmission in one unit of time are p, the total packets lost are $p \times t$. Putting these values together, $f(\beta) = \beta\rho\pi R^2 pt$.

Another example is when the attacker tries to compromise the measurement integrity by sending wrong sensor measurements to the controller. Assume that the attacker has compromised one node and that measurements are taken periodically at times $t = k\Delta t$ for $k = 0, 1, \ldots$, where Δt is a time interval. In this scenario, $f(\alpha) = \sum_k \parallel x(k\Delta t) - x'(k\Delta t) \parallel^2$. In our SCADA system example, the sensor reading x might correspond to the fluid level in

a tank. In this case, the attacker can modify the readings to be $x' = x + \xi$ where ξ can be a positive or a negative value. There is no restriction on how the attacker chooses ξ. However, if he wants to affect on the decision making process of the system, he has to choose ξ intelligently so that the resulting x' is a valid fluid level, although it is not the *correct* fluid level.

3.3.6 Ranking

In this section, we present a ranking of various attacks in terms of their effect on availability and measurement integrity. The ranking is based on the difficulty of performing the attack versus the affect it has on the corresponding security objective. The effect of each attack is captured by the attacker's objective function defined in Equation 3.3.5.

These rankings are based on the assumption that there are no security mechanisms in place. Therefore spoofing for example, is a very devastating attack, because an attacker can impersonate any node and send arbitrary data. We also assume there is no physical security, so launching one of our outside-physical attacks is assumed to be very easy for an attacker.

Having no security mechanisms in place for this analysis will give us a base line for attack rankings which can be used to decide what security mechanisms are most effective in preventing each attack.

In order to better quantify the feasibility of each attack, we define a threat ranking which gives an overall score of how difficult it is to accomplish the attack. We make a number of assumptions which are outlined below:

1. Intermediate attacks are more difficult to launch than similar final attacks.

2. Outsider attacks are easier to perform than insider attacks.

3. For general insider attacks, the attacker has control of one node but does not spoof any other node.

4. To launch a successful Sybil or Wormhole attack, the adversary has at least two identities.

5. Having at least two identities is more difficult to accomplish than a single identity.

In addition to the above assumptions, we consider the following points when quantifying each attack:

- Cost of extra hardware
  - Attack requires more than commodity hardware.
  - No extra hardware required, i.e. the attack can be accomplished using PC, laptops and sensor motes.
- Physical access: refers to physical proximity to the sensors and being able to touch them.
- Required technical skill
  - Brute-force attacks: (e.g. destroying a node, or jamming a network).
  - Logical attacks: these attacks require the knowledge of the specific protocol being used by the network.
    * Automated attacks: the attack scripts can be acquired easily, such as, through the Internet.
    * Non-automated attacks: The attacker needs to invest more amount of time and resources for these attacks.

Figures 3.2 and 3.3 show a possible ranking of the service-integrity attacks. Our attack ranking has a partial ordering because we strongly believe that some attacks are not comparable. For example, an environment tampering attack may be very easy to perform in some cases, but if a sensor is installed in a pipe or a water tank, the attack may be more difficult to launch than physically compromising a node to send fake data.

To have a better intuition of which attacks should be the first priority of the network user, we have used an arbitrary representation of the partial order into a total order in Figure 3.4. We have performed a similar analysis for availability, which is shown in Figures 3.5, 3.6, and 3.7.

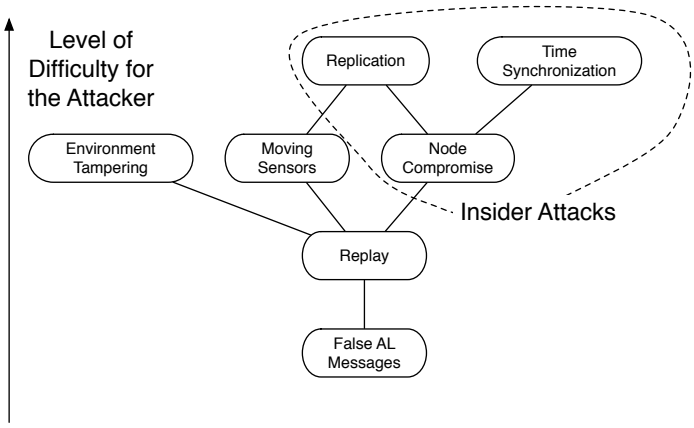


Figure 3.2. This figure shows a partial order for the difficulty of performing each attack. These attacks are concerned with data integrity. False AL refers to false application-layer packets, and assumes a successful message insertion, message tampering, or spoofing attack.

3.4 The Design Space

The majority of research in the security of sensor networks has focused on implementing security mechanisms with the assumption of severe resource constraints and no online trusted third party. While this scenario covers a large class of practical sensor networks, it is important to realize that these are not the only sensor networks available. Sensor networks have been used for a wide variety of applications and systems with vastly varying require-

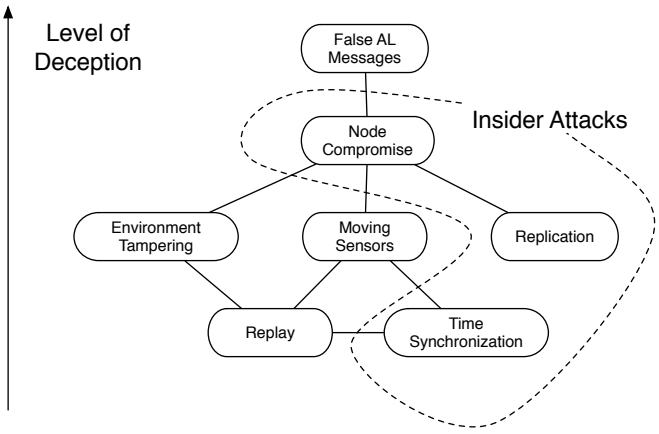


Figure 3.3. Level of deception, or compromise in the data integrity, when each attack in this figure is carried out.

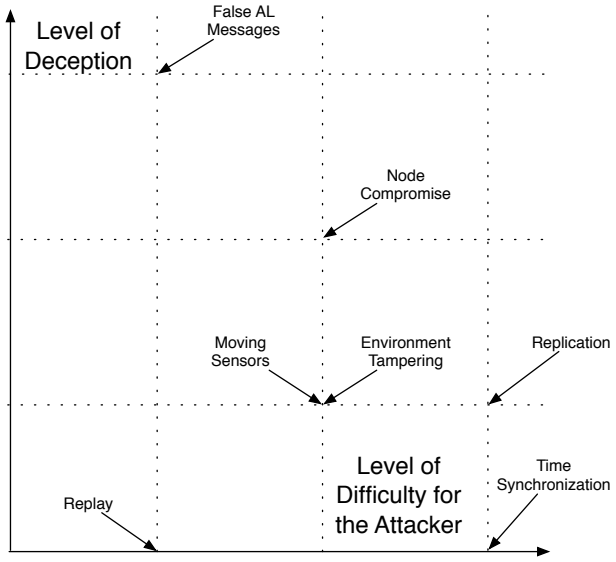


Figure 3.4. A 2-D graph of the difficulty of each attack versus its consequence on the data integrity. We can see that the false-application later message attack is relatively easy to carry out and has a high impact on the integrity.

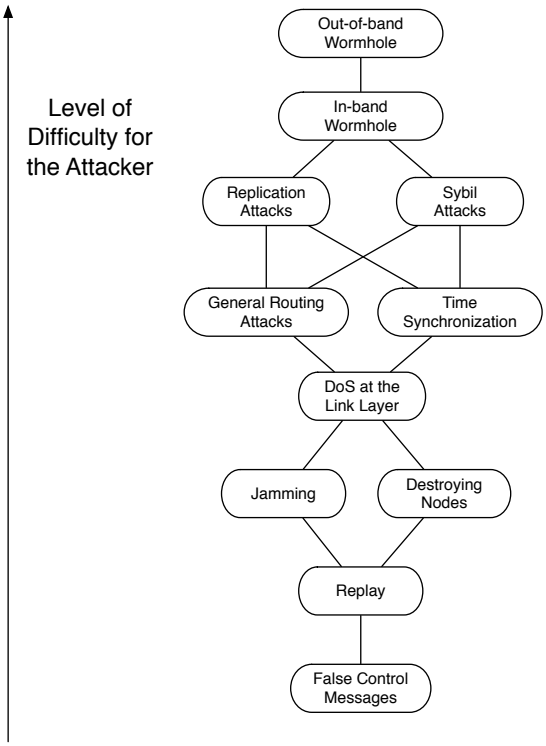


Figure 3.5. Partial order for the difficulty of performing DoS attacks. DoS attacks impact the availability of the network. False control messages refer to attacks where the adversary can fake, spoof or tamper

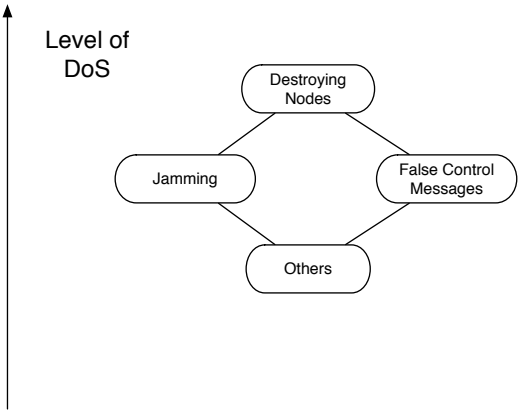


Figure 3.6. The consequences of carrying out DoS attacks on the network availability.

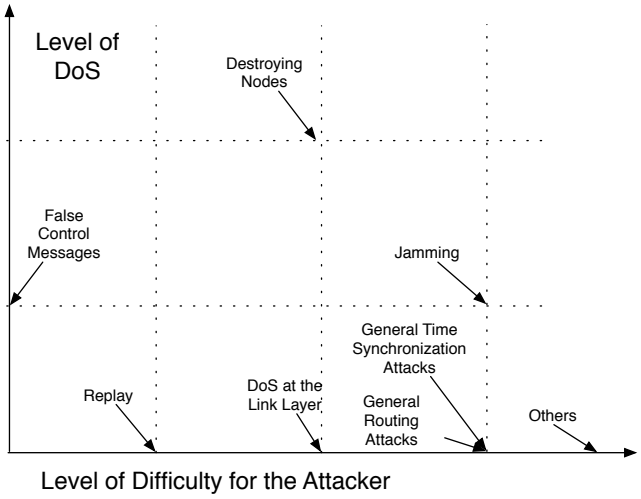


Figure 3.7. A 2-D graph that illustrates the difficulty of carrying out each attack and its corresponding impact on the availability of the network. The figure indicates that if there is no physical security, there is no point in adding, for example, a secure routing protocol, because the adversary will destroy the nodes.

ments and characteristics. In a recent study [68], the authors show the diversity of sensor networks in terms of deployment, cost, size, resources, energy, heterogeneity of the sensor nodes, infrastructure, size, lifetime, and other considerations.

We favor considering a more diverse design space due to the following:

- There is no global definition of what a sensor network is, meaning, each sensor network deployment will have its own goals, requirements, and constraints.
- The traditional assumptions about sensor networks might change radically as sensor networks start being deployed in practical commercial applications.
- The security advantages that we obtain by investing in a more expensive infrastructure might outweigh the costs.
- Attempting to add too many security countermeasures to a resource constraint sensor network will overwhelm the network and make the troubleshooting more complex.

We now discuss the advantages and disadvantages of some security mechanisms. The

appropriate security mechanisms have to be chosen based on the importance of each security requirement and the ease of lunching various attacks to compromise each security objective. This means that one must secure the path of least resistance in the network first. This path can be found by considering the figures in Section 3.3.6. Each figure illustrates the ease, or alternatively difficulty, of launching each attack versus the impact it has on the network functionality. Therefore, in order to secure a sensor network, one has to start by using mechanisms that prevent attacks with less difficulty but more impact. Examples of this type of attack are the replay attack or spoofing.

3.4.1 Types of Security Mechanisms

The basic security mechanisms can be divided into three parts: 1) prevention, 2) detection, and 3) survivability. We explain each part in more details:

Prevention mechanisms typically rely on cryptographic algorithms implementing authentication and access control. These mechanisms prevent the attacker from participating in the communication and compromising integrity.

Detection relies on placing additional sensors for detecting unusual activities. We believe detection is most useful if the sensor node that is compromised is able to report this action. If the sensor node cannot report the compromise, distributed detection mechanisms have been proposed, such as, watchdog monitoring, or distributed revocation. These distributed protocols open a new class of problems, such as, *slander* attacks where compromised nodes can accuse good nodes of malicious behavior.

Survivability is the ability of the network to remain in operation despite attacks on the system.

In the next section we are going to discuss key management in more detail since it is the

essential building block for most solutions. Then we will continue with the details of each of the security mechanisms discussed above.

3.4.2 Key Management

There are three types of key agreement schemes: schemes with an online trusted server, public-key schemes, and key pre-distribution schemes. In order to better understand each scheme, we outline the advantages and disadvantages of each scheme in what follows.

Key Pre-distribution Schemes [32]–[34], [69]:

- Advantages: Key pre-distribution is based solely on symmetric cryptography. Furthermore, it does not rely on an online trusted third party (the trusted party is only used for the offline pre-distribution of keys).
- Disadvantages: It is difficult to guarantee end-to-end security among nodes. Furthermore, since there are repeated keys among several nodes, an adversary that compromises a few nodes can compromise the confidentiality and message integrity of a larger part of the network, and not only of the nodes it has just compromised (in its simplest case, key predistribution will deploy a single key for the whole network, and the compromise of the key will affect communications between all nodes.)

Online Trusted Server Schemes [67]:

- Advantages: Online trusted server schemes rely solely on symmetric cryptography: each node shares a unique secret key with an online key distribution center. It is more resilient to node compromise than key pre-distribution, since the compromise of n nodes compromises the security of only these n nodes.
- Disadvantages: The network relies on the availability and integrity of the trusted server. If the trusted server is made unavailable, the sensor network would not be able to oper-

ate securely. Furthermore, if an adversary compromises the trusted server, the security of the sensor network is compromised (notice however that the same problem would occur if the offline trusted party for key pre-distribution schemes or the certificate authority for public-key schemes is compromised).

Public-key Schemes [34]:

- Advantages: Public-key schemes do not rely on an online trusted third party although the trusted third party can function as an offline certificate authority. This schemes are more resilient to node compromise when compared to key pre-distribution schemes. Furthermore, the public key of the device binds its identity to the network. This is useful for authenticated one-to-many communications, such as, broadcast. Even with some compromised nodes in the network, a one-to-many message signed by the source of the message can authenticate its origin. Protocols like μTESLA [4] can achieve this property with symmetric-key primitives, but they incur in authentication delays and require tight time-synchronization.

- Disadvantages: Public-key schemes are based on asymmetric-key algorithms, and although they are now assumed to be feasible in most sensor network infrastructures [70], [71] their use would still deplete the battery of sensor nodes faster than symmetric- key operations. Asymmetric cryptography, however, is typically used only to establish symmetric session keys, thus the influence to the lifetime of the network might not be significant to the new generation of sensor nodes [72]. In particular, the use of elliptic curve cryptography (ECC) and hardware support can help improve the efficiency of asymmetric algorithms.

The key management protocol provides the nodes in the sensor network with the following cryptographic keys:

1. **Network key** shared by all authorized entities,

2. **End-to-end keys** shared only between principals communicating at the application layer of the network, and

3. **Pairwise link-layer keys** shared between neighboring nodes at the link-layer of the network.

Key pre-distribution schemes have been studied extensively in the research literature, mostly motivated by the seminal paper of Eschenauer and Gligor [33]. In sensor networks, the availability of a trusted server in many practical scenarios has motivated a number of standards, such as, the ZigBee alliance [73], ISA SP100 [55] and WirelessHart [54], to propose the use of an online trusted network managers for secure networks. The use of public key cryptography is also being supported by the standards associations and several other companies, such as, NTRU's Aerolink [74].

3.4.3 Confidentiality Mechanisms

Encryption is the primary way of preserving the confidentiality of the data packets in a network. However, one has to choose an encryption technology that is resilient to chosen-plaintext attacks (semantic security) since it is infeasible for a computationally-bounded adversary to derive any significant information about a message when he is only given its ciphertext. A few points to consider when deciding on what encryption mechanism to use are:

- A single network key is not resilient to insider attacks.

- A countermeasure for insider attacks on confidentiality is to use pairwise link-layer keys. If the adversary compromises the key of one node, it can only eavesdrop on communications passing through this node.

- A stronger guarantee is to use end-to-end encryption. If end-to-end encryption is the only encryption method used, any information below the application layer is disclosed (routing information, link layer addresses etc.). Therefore, it might be necessary to use end-to-end encryption with pairwise link-layer keys, or a single network key used at the link layer. End-to-end encryption might limit the use of distributed protocols, such as aggregation schemes [5], [46]. This essentially becomes a question of the tradeoff between confidentiality and energy efficiency.

Decryption of the message is not the only way in which an outsider can infer the contents of the message. Monitoring of the environment can be **polled-based, periodic, or event driven**, as mentioned in Chapter 2. A typical event-driven monitoring application uses *alarms*. A sensor will only send a report if an event is detected. Therefore, an eavesdropper can identify that an event has happened if he observes the sensor node sending a packet. In addition, sensor nodes typically send health reports, such as, battery status or wireless link connectivity, back to the base station. Therefore, a possible countermeasure to mitigate these attacks is to randomize the time in which sensor nodes contact the base station. In order for this measure to be successful, we have to ensure that the eavesdropper cannot distinguish between encrypted health reports and encrypted alarms. Finally, an adversary might be able to place its own sensors for monitoring the environment. This is not a technical attack, but it shows the importance of protecting the physical deployment area of the sensor network. A possible mechanism to detect and deter this attack is to use surveillance cameras.

3.4.4 Service Integrity Mechanisms

The typical way to provide end-to-end data integrity, data liveness, and data origin authentication is to include a message integrity code (MIC), also known as message authentication code (MAC), in the packets sent by each party. The MIC should include,

- The identities of the communicating parties at the application layer for data origin authentication
- The sensor reading for data integrity
- A counter for *weak* liveness, a time-stamp for *strong* liveness, or a nonce for *strong* liveness if the data is polled and the requesting party sends the nonce in the request. For polled messages, the nonce would provide the stronger guarantee of data liveness, since the time stamp depends on accurate time synchronization.

Similar to the confidentiality, a network key for MIC gets broken when an adversary compromises a single key. Therefore, using the end-to-end key guarantees that the message can not be tampered with even if the attacker has compromised the keys of other nodes in the network. This solution comes at the cost of limiting the use of distributed aggregation algorithms. The use of pairwise-link layer keys will also limit the effect of insider attacks when attacking the time-synchronization protocol. As a result, we can use the time stamps for determining the data liveness.

Measurement integrity can be protected to some extent by tamper-resistant or tamper-detection hardware. This increases the effort the adversary needs to put in to compromise the sensor node. The node might also include external sensors to detect when it is being moved to another location. Finally, a way to prevent or detect 'environment attacks', such as, placing a magnet on top of a magnetometer, is to attempt to protect the physical area of the sensor node by using surveillance cameras.

Other possibility for *detecting* and *surviving* a measurement integrity attack is to use robust statistics [48], and of particular importance to SCADA systems, the use of robust control [75]. By identifying outliers and anomalies in the messages received, the measurement integrity attack can be limited. Robust statistics and robust control come at a cost: even if there is no attack, they might discard true anomalous information.

3.4.5 Availability Mechanisms

The data packets transmitted throughout the network can be divided into **control packets** and **application packets**. Application packets are packets whose payload contains data sent by the application layer of the network, such as sensor readings. Up to this point, our focus has been on protecting the integrity and confidentiality of application packets. On the other hand, there are packets whose payload contains data used to maintain the network services. These include: routing discovery packets, routing maintenance packets, time synchronization packets, etc. All of these control packets are of fundamental importance for availability of the network.

Contrary to confidentiality and integrity, we cannot assume end-to-end security in network availability because the network is a distributed entity. Therefore, we can only use a network key or a pairwise link-layer key. The same principles of service integrity and service confidentiality apply to control packets. To provide a secure communication infrastructure, control packets need to have authentication and replay protection. Therefore, these packets need to use message integrity codes. Also, to prevent an outsider from identifying the type of control packets being transmitted, we should have these packets encrypted. Without any key, an outsider can attempt to perform jamming or physical destruction of the sensing nodes. Other outsider attacks, such as replay, spoofing, and even a wormhole will have much limited effect if the the adversary cannot identify control packets and their payloads.

The design space for jamming and its countermeasures is highly situation dependent. Military sensor networks might have a very large design space for countermeasures, such as: (1) prevention: implementation of advanced waveforms using spread spectrum techniques for low probability of detection, and low probability of intercept. (2) Survivability: dynamic frequency reallocation, and raising the transmit power. However, commercial sensor networks cannot have the same flexibility in their design space because they have to conform to several norms. For example, in the U.S., commercial wireless systems have to be approved by the

federal communications commission (FCC). The use of frequency hopping spread spectrum might make jamming more difficult to the adversary, but it will not prevent the jamming attack all together. If the adversary has physical access to the sensor network it can also destroy the nodes. One way to discourage an adversary from performing jamming is to increase the physical security defenses to the sensor network deployment field.

The availability of the network can be compromised if the attacker is able to gain access to the keying materials. To prevent an adversary from compromising the operation of all the network by capturing a single key, pairwise link-layer keys among neighboring nodes must be used. Under these circumstances an adversary becomes a Byzantine attacker who can try to disrupt the network operation by 'confusing' peer devices. Therefore, the resiliency of the network management protocols will rely on the redundancy of resources. Another area that has attracted a lot of attention is the security of the routing protocols. Disrupting the functionality of the routing protocols compromises the availability of the network and services running on the network. Attacks against routing protocols include attempts to create routing loops and black holes. We can identify two possible countermeasures: (1) measuring the link quality based on the number of dropped packets, and (2) using a routing protocol that builds path diversity. If a message sent along one path is not acknowledged by the recipient, then the protocol should use alternative path via a different neighbor. This could also mean using multi-path routing protocols that send the data along different paths and take advantage of the redundancy in the received data.

3.5 Discussion

In this chapter, we presented the first steps towards a holistic and systematic view of the security of sensor networks. We believe this research direction will provide a better understanding of the security issues and will help the network designer decide on the most effective security mechanisms under resource constraints. However, there are many research

challenges that remain, such as, developing a systematic analysis of the threat model and its relation to the security countermeasures, the precise definitions of *security metrics*, and the detailed study of real world deployment scenarios.

Part 2

Protocol Security

Chapter 4

Time Synchronization Security

Time synchronization protocols provide a mechanism for synchronizing the local clocks of the nodes in a sensor network. There are several time synchronization protocols for the internet, such as Network Time Protocol (NTP). However, given the non-determinism in transmissions in sensor networks, NTP cannot be directly used in wireless sensor networks. Time synchronization implementations have been developed specifically for sensor networks. Three of the most prominent are Reference Broadcast Synchronization (RBS) [2] Timing-sync Protocol for Sensor Networks (TPSN) [51], and Flooding Time Synchronization Protocol (FTSP) [52]. However, none of these protocols were designed with security as one the goals[1][2].

In this chapter we focus on insider attacks on time synchronization protocols. We assume that the adversary physically captures a node and gains access to the cryptographic keys on the captured node. This will enable him to participate in the authenticated communication with other sensor nodes without being recognized as an attacker. In the context of time synchronization attacks, the adversary compromises a node and injects erroneous time

[1] Permission to use parts of these works for the purposes of this dissertation have been granted by the IEEE and/or the appropriate copyright holder(s).

[2] The contents of this chapter are based on [59], [60].

synchronization information in the network. We look at the affect of this type of attack on various applications. In addition, we present the result of our testbed implementation of the attacks and countermeasures.

The rest of the chapter is organized as follows. Section 4.1 discusses the clock model and the communication model used in our discussions of time synchronization protocols. Section 4.2 gives the details of the operation of the three main time synchronization protocols in sensor networks. Section 4.3 discusses the attacks on each group, followed by the effects of the attacks on various applications that use time synchronization in Section 4.4. Section 4.5 discusses our proposed countermeasures for the attacks. Finally, Section 4.6 discusses our experiments on implementing the attacks and some countermeasures for the FTSP protocol.

4.1 System Model

In this section, we define the problem of secure time synchronization and discuss the clock model, communication model, trust assumptions, threat and attacker models, and security goals we wish to accomplish.

4.1.1 Clock Model

Every sensor node has a notion of time that is based on the oscillation of a quartz crystal. The sensor clock has a counter that is incremented at rate f, where f is the frequency of the oscillation. The counter counts time steps, and the length of these time steps is defined by the system designer. The clock estimates the real time $T(t)$, where,

$$T(t) = k \int_{t_0}^{t} \omega(\tau) d\tau + T(t_0) \tag{4.1}$$

$\omega(\tau)$ is the frequency of the crystal oscillation and k is a constant [76]. Ideally this frequency should be 1, i.e. $\frac{d\omega}{dt} = 1$. However, in reality the frequency of a clock fluctuates over time due

to changes in temperature, pressure, and voltage. This will result in a frequency different than 1. This difference is termed *clock drift*. There are a number of ways to model the clock drift. In addition to frequency fluctuation in one clock, the crystals of different clocks oscillate at different rates. This difference causes what is called the *offset* between two clocks [76].

4.1.2 Communication Model

In order to perform time synchronization, the sensor nodes send time synchronization messages to each other. The messages are sent through wireless channel. We assume that the wireless links are symmetric, meaning if node A hears node B, then node B also can communicate with node A. However, it is worth mentioning that in reality the wireless links are not always symmetric; in fact, it has been shown through experiments that the wireless channel is asymmetric.

4.1.3 Adversary Model

In sensor networks there are one or more base stations, which are sinks and aggregation points for the information gathered by the nodes. Since base stations are often connected to a larger and less resource-constrained network, we assume a base station is trustworthy as long as it is available. Beside the base stations, we do not place any trust requirements on the sensor nodes because they are vulnerable to physical capture.
While it is possible that the adversary has access to sensor nodes very identical to the ones deployed or has more powerful nodes such as laptops, here we only consider a mote-class adversary.

An outsider attacker has no special access to the sensor network, such as passive eavesdropping, but an insider attacker has access to the encryption keys or other code used by the

network. We consider only an insider attacker. We assume that the adversary has been able to capture and corrupt a fraction of the total nodes in the network. The adversary, therefore, has access to the secret keys for authorized communication with other nodes. The goal of the adversary in our setting is to inject false time synchronization information in the network, without being detected by the honest nodes.

4.2 Time Synchronization Protocols for Sensor Networks

Sensor networks are used for monitoring different real world phenomena. Since the existing time synchronization protocols do not fit the special needs of sensor network, a number of clock synchronization protocols have been developed to meet the memory and energy constraint of these networks. There are three main ways to synchronize nodes together. In the first approach an intermediate node is used to synchronize the clocks of two nodes together, such as Reference Broadcast Synchronization [2]. The second approach assumes that the clock drift and offsets are linear, and nodes perform pair-wise synchronization to find their respective drift and offset, such as TPSN [51]. In the third approach, one node declares itself the leader, and all the other nodes in the network synchronize their clocks to the leader, such as Flooding Time Synchronization Protocol [52]. In the following sections, we review these three approaches to the problem of time synchronization in sensor networks. For a comprehensive survey of clock synchronization protocols for sensor networks we refer the reader to [77].

4.2.1 Reference Broadcast Synchronization (RBS)

In RBS a reference message is broadcast to two receivers and the receivers synchronize their respective local clocks to each other, Figure 4.1. Each receiver records its local time when it gets the reference message. Then the two receivers exchange their local times. It is

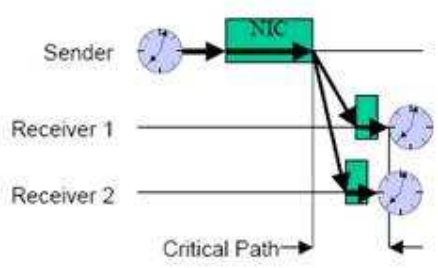


Figure 4.1. RBS does not synchronize the receivers to the sender. Instead it synchronizes the receivers to each other. Figure courtesy of [2].

possible to increase the precision of the estimated time by broadcasting m reference messages instead of one, as follows [2]:

1. A transmitter broadcasts m reference messages.
2. Each of the n receivers record the time they received the reference messages based on their respective local times.
3. The receivers exchange their local times.
4. Each receiver can calculate its phase offset as the LS linear regression of the phase offsets implied by each reference message observed by the receivers as follows: Each node finds the time difference between itself and a neighbor for each reference broadcast message. Then the node finds the least-squares (LS) error fit for these points. The node can then find the offset and skew of its clock from the slope and intercepts point of the line.

The advantage of RBS is that it eliminates the non-determinism on the transmitter side. RBS does not use a multi-hop scheme to synchronize all the nodes to the same global clock. For the multihop case, the nodes that fall within the overlapping region of two reference transmitters perform clock conversion, to go from one clock estimate in one region to the other region. By doing the clock conversion, it is possible to find the time difference between events that happen at different parts of the network.

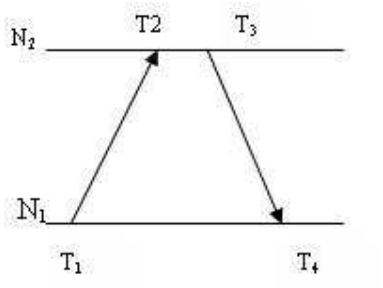


Figure 4.2. An example of TPSN

4.2.2 Time-Sync Protocol for Sensor Networks (TPSN)

TPSN initially creates a spanning tree of the sensor network. The tree starts at the root of the network, which is generally a base station, and each node establishes its level based on the 'level-discovery' message that it receives. While the tree is being built, the nodes perform pairwise synchronization along the edges of the tree. Each node exchanges synchronization messages with its parent in the spanning tree. By comparing the reception time of the packets with the time of transmission, which is placed on the packet by the parent, the node can find and correct for its own phase offset. We give an example to clarify this scheme. Let us assume there are two nodes, N_1 and N_2, where N_1 is the parent of N_2, Figure 4.2. When N_2 wants to synchronize its clock, it sends a *synchronization pulse* to N_1 along with the value T_1, the time the packet is transmitted from N_2.

When N_1 receives this packet, it sends back an *acknowledgement* packet with times T_2, reception time, and T_3, retransmission time. N_2 receives this ACK packet at T_4, and using the four time values it can find the clocks drift and propagation delay, using the equations [51] $d = \frac{((T_2-T_1)+(T_4-T_3))}{2}$.

TPSN is a sender-initiated time-synchronization protocol, and the sender synchronizes to the receiver's clock. TPSN has a better performance than RBS by time stamping the messages at the MAC layer of the radio stack. It also uses a two way message exchange instead of a one way exchange as in RBS [51]. However, TPSN does not account for clock drift. It also

does not efficiently handle dynamic topology changes because it has to compute the spanning tree of the network every time a change happens.

4.2.3 Flooding Time Synchronization Protocol (FTSP)

In FTSP, a root node broadcasts its local time and any nodes that receive that time synchronize their clocks to that time. The broadcasted synchronization messages consist of three relevant fields: *rootID*, *seqNum*, and *sendingTime* (the global time of the sender at the transmission time). Upon receiving a message, a node calculates the offset of its global time from the global time of the sender embedded in the message [52]. The receiving node calculates its clock skew using linear regression on a set of these offsets versus the time of reception of the messages.

Given the limited computational and memory resources of a sensor node, it can only keep a small number of reference points. Therefore, the linear regression is performed only on a small subset of the received nodes. Since this regression requires that set of updates, however, a node cannot calculate its clock skew until it receives a full of reference messages. Therefore, there is a non-negligible initiation period for the network.

FTSP also provides multi-hop time synchronization in the following manner: Whenever a node receives a message from the root node, it updates its global time. In addition, it broadcasts its own global time to its neighbors. All nodes act in a similar manner, receiving updates and broadcasting their own global time to their neighbors. To avoid using redundant messages in the linear regression described above, each node retains the highest sequence number it has received and the *rootID* of the last received message used. A synchronization message is only used in the regression if the *seqNum* field of the message (the sequence number of the flood associated with that message) is greater than the highest sequence number received thus far and the *rootID* of the new message (the origin of the flood associated with that message) is less than or equal to the last received *rootID*. FTSP is more robust against node failures and

topology changes than TPSN since no topology is maintained and the algorithm can adapt to the failure of a root node. If a node does not hear a time synchronization message for a ROOT_TIMEOUT period, it declares itself to be the new root. To make sure there is only one root in the network, if a root hears a time synchronization message from another root with lower ID than itself, it gives up its root status.

4.3 Attacks on Time Synchronization Protocols

In this section, we discuss different attacks on the time-synchronization protocols explained above. We discuss possible attacks on RBS, TSPN, and FTSP in each subsection. All the attacks on time synchronization protocols have one main goal, to somehow convince some nodes that their neighbors' clocks are at a different time than they actually are. Since global time synchronization is build upon synchronization at the neighborhood level, this will disrupt the mechanisms by which the protocols above maintain global time in the network or allow events at distant points in the network to be given to be give time-stamps which reflect the actual difference between their times of occurrence.

4.3.1 Attacks on RBS

As mentioned above, in RBS a reference message is broadcast by the base station. Two nodes that receive this message exchange their local time for clock synchronization. However, if a node is compromised, it can send a falsified synchronization message to its neighbor during this exchange period. The effect is that the honest node will calculate an incorrect phase offset and skew. In the field of Robust Estimation, the *breakdown point* of an estimator is defined as the smallest fraction of contamination in data that can cause the estimator to take on values arbitrarily away from the real value. It is known that the average and LS estimators have a very low breakdown point. For example, the breakdown point for the LS estimator

is $\frac{1}{n}$. Therefore, as n gets larger, the breakdown point gets closer to 0, meaning that a very small fraction of contaminated data can cause the estimation to get far from the real value. If the number of nodes is large in a sensor network, even one compromised sensor node can cause the time synchronization estimates to get far from the true global time.

It is also possible to attack the multi-hop version of RBS. As mentioned before, RBS does not keep a global time; instead, it tries to find the time difference between the occurrences of two events at different parts of the network. Since the nodes on the boundary of the overlapping regions perform clock conversion, a compromised node placed in any of the overlapping regions can affect multiple regions by giving an erroneous value in the clock conversion. Once the adversary sends a miscalculated clock conversion, this error will be propagated as it is sent throughout the network.

4.3.2 Attacks on TPSN

TPSN creates a spanning tree of the network at the initial phase of level-discovery and then performs pair-wise clock synchronization. As mentioned above, TPSN is a sender-based synchronization protocol, so the adversary can only affect the protocol through manipulating its children in the spanning tree. That is, the adversary will not be able to cause any disruption by initiating a time-synchronization request since only the parent node will reply. However, the compromised node can effect its children by sending incorrect time stamps for the reception time and the transmit time of the time-synchronization request. The adversary can propagate this de-synchronization down the spanning tree toward the leaves on its own branch. Therefore, the closer the compromised node is to the root, the more effect it has on time synchronization in the network, and the larger a region it can contaminate. Alternatively, the adversary can place more compromised nodes at different branches of the tree, and affect a wider region of the network.

The compromised node can also lie about its level in the tree, meaning it can claim a lower

level than its real level. By doing so, it can convince the other nodes at its level to request time-synchronization updates from the compromised node. In addition, the compromised node can avoid participating in the tree building phase, and by doing so cut off a number of nodes that would have been its children from the root. If those nodes cannot find any other node as their parent, they will not be able to synchronize to the rest of the network. The effects of this attack would be greatest in a sparse network topology.

4.3.3 Attacks on FTSP

The major innovations of FTSP, in terms of multi-hop synchronization, over RBS are that the root is chosen dynamically and any node may claim to be the root if it has not heard time updates for a preset interval. One possible attack on this protocol is for the compromised node to claim to be the root node with ID 0 and begin at a higher sequence number than the actual root node so all the updates originating at the actual root node will be ignored. This can be easily done since the protocol allows any node to elect itself root to handle the situation where nodes have not heard from the root node for a long period. Once a compromised node becomes the root, it can give false updates to its neighbors, which will in turn propagate that false time to their neighbors and so on. Every node that accepts the false updates will calculate a false offset and skew for its clock. A more comprehensive treatment of attacks on FTSP along with testbed implementation of the attacks is presented in Section 4.6.

4.4 Effects of Time Synchronization Attacks on Sensor Network Applications

To motivate our discussion of time-synchronization attacks, we describe here in detail the effects of time synchronization attacks on a set of sensor network applications and services that are dependant on time synchronization.

In many application areas, time synchronization allows engineers to design simpler and more elegant algorithms. If security is a high priority, however, the simplest countermeasure against an attack on time synchronization is to build the algorithm such that it does not rely on a time-synchronization service whenever possible.
That said, for certain classes of applications under certain conditions, algorithms cannot provide correct results without an accurate and reliable time-synchronization service. Rather than try to describe those conditions, we have tried to select a representative set of applications that rely on a time synchronization service and demonstrate the effects on them of a time synchronization attack. While we believe these algorithms to be representative of the set of algorithms relying on time synchronization, the set is not exhaustive.

4.4.1 Shooter Localization

The shooter localization algorithm designed by Vanderbilt University [3] has three key stages:

- Detection: To find the location of the sound of the muzzle blast, the nodes report the time of arrival (TOA) of the sound on three audio frequencies. The sound is processed to determine whether it will be reported to the base station. The report contains an age field.

- Routing-Integrated Time Synchronization: In this step, the Directed Flooding Framework is used. As the message is routed toward the base station, the routing nodes update the age field by adding the elapsed time between reception and retransmission of the packet. MAC layer time stamping is used, as in FTSP, to update the age field precisely. This age field is used at the base station to find the exact time of the shot [3].

- Sensor Fusion: Once the muzzle blast TOA data is submitted to the base station, a

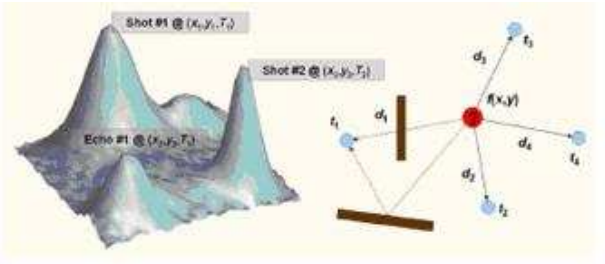


Figure 4.3. Consistency function to determine the time and location of the sound from a gunshot [3]

sensor fusion algorithm is used to estimate the shooter location and the trajectory of the projectile.

The fusion algorithm searches for the maximum of a consistency function, which corresponds to the location of the muzzle blast. The consistency function is defined on four-dimensional space-time space as follows: let (x, y, z) be the shooter position and let shot time be t. The theoretical time of arrival of the shot at the sensor that recorded the ith measurement is t_i and is calculated as follows [3]:

$$t_i(x, y, z, t) = t + \frac{\sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}}{\nu} \tag{4.2}$$

Where ν is the speed of sound, and (x_i, y_i, z_i) is the coordinates of the sensor which is making the ith measurement. If the sensors were in a line with the shot, then the real shot time and the estimated time should be equal. However, in reality this does not happen. As a result, the considered measurements are those that satisfy:

$$|t_i(x, y, z, t) - t_i| \leq \tau \tag{4.3}$$

where t_i is the actual time of arrival of the shot at sensor i, $\tau = \frac{\delta_1}{\nu} + \tau_2 + \tau_3$ is the uncertainty value, δ_1 is the maximum localization error, τ_2 is the maximum time synchronization error, and τ_3 is the maximum allowed signal detection uncertainty. It is usually assumed that the uncertainty value is dominated by the localization error and not the other two terms. There-

fore, from the above definition the value of the consistency function is defined as the number of observations that support the claim that the shot was taken from location (x, y, z) at time t with uncertainty τ, where it is assumed that an upper bound is known for τ based on the localization, time synchronization and the signal detection algorithms :

$$C_\tau(x, y, z, t) = count_{i=1,\ldots,N}(|t_i(x, y, z, t) - t_i| \leq \tau) \tag{4.4}$$

The maximum of this consistency function gives the location and time of the shot, as shown in Figure 4.3. The maximum number of observations if found by using a sliding window on the time line, and counting the number of sensor observations at each step of sliding the window. The size of the sliding window is on the order of milliseconds.

However, if the time synchronization protocol has been attacked by compromised nodes, the consistency function can be affected. The maximum time synchronization error might become comparable to the localization error. This might change the shape of the consistency function and, consequently, change the value and location of the local maximum. This would yield an incorrect estimate of the location and time of the shot. For example, if the consistency function is as in Figure 4.3, there are three local maxima in the consistency function, meaning there were three time slots where the count was high. If, due to a time synchronization attack, some of the nodes that reported time t_1 for the first shot report a shot at $t_1+\Delta$, and Δ is greater than the width of the sliding window, the first maximum will be pushed down and the location and time of the shot will incorrectly be estimated at another peak in the consistency function. This means that the location of the shooter will be reported in a different location than its true position or not reported at all if the shot time falls in a portion of the consistency function that is not a local maxima. However, this requires that the nodes in the neighborhood of the shooter all be desynchronized by the same amount Δ, and the value of Δ is on average as large as the time slots used by the administrator to count the report. Also, if the clocks of the sensor nodes in a neighborhood are out of sync with each other, they send their reports

at different times (absolute time). When the base station processes the received reports, the counts could potentially fall in different time slots, and instead of having a maximum in one time slot, there is no maximum. As a result, the location of the shooter can not be determined. In short, the amount by which the localization is going to be thrown off is proportional to the ratio of the dysynchronization error Δ to the uncertainty τ.

4.4.2 TDMA-based Channel Sharing

In TDMA-based channel sharing protocols, time is divided into intervals and each node is assigned a schedule in which to transmit and receive messages. In this section, we are going to discuss two different protocols that use TDMA-based approaches. The first is the flexible power scheduling [78] and the second is *PEDAMACS* [79].

Flexible Power Scheduling

The goal of this protocol is to reserve time slots for transmission and reception of data messages on each node so that the nodes can go to sleep during the idle slots. This saves power because nodes do not have to be listening to transmissions the entire time. We give an outline of the protocol below:
Time is broken into cycles and each cycle consists of m time slots. During each slot the node has one of the following six states [78]:

- *Transmit (T)*: The node may transmit a message to its parent.
- *Receive (R)*: The node may receive a message from its child.
- *Advertisement (A)*: The parent node broadcasts an advertisement to find an available reservation slot from the child
- *Transmit Pending (TP)*: The node sends a reservation request to its parent.

- *Receive Pending (RP)*: The node receives a reservation request from its child.
- *Idle (I)*: The node powers down during this period if this slot in its schedule is unoccupied.

The algorithm proceeds as follows: When a node is in an A slot, its parent advertises for a reservation slot and marks the corresponding slot in its own schedule as RP. If a child node hears this advertisement, but still has not met its demand, it marks the corresponding slot in its schedule as TP, and sends a reservation request when the time slot arrives. To initialize the algorithm, the base station picks a reservation slot at random and sends out an advertisement for it. It then waits for a reply during the following cycle. When a new node joins the network, it sets all slots in its schedule to I and then listens for one cycle for an advertisement. In order to keep the time synchronized, once a node selects a parent, it periodically synchronizes its time to the parent. This is similar to TPSN in the sense that the child synchronizes its clock to the parent in a spanning tree of the network.
We can summarize the FPS algorithm's main steps during each cycle in the following [78]: After initialization, each node picks an advertisement slot A randomly from the idle slots. Then it picks a reservation slot (RP) at random from the idle slots.
If supply $\geq$ demand,

- Turn off the radio during the idle slots
- Schedule an advertisement during an A slot

Otherwise,

- Leave the radio on and listen for advertisement Schedule a reservation request during a TP slot

For each time slot, the node checks its power schedule, and performs an action according to its schedule. For example, if the slot is marked T, it transmits a message. At the end of the cycle, the node clears the current slot and the previous RP from the schedule.
If the compromised node is a parent, it can send wrong time synchronization messages, and that will cause the schedule of the child to be inconsistent with the parent's schedule, i.e. the time slots corresponding to the same action in the two schedules will not be aligned in time. Since the parent can send wrong messages to different children, the time slots in multiple nodes will conflict with each other, and as a result, there will be more collisions due to transmission at overlapping time slots. Even though the protocol does not need a fine-grained time synchronization protocol to work, when the error in local clocks gets accumulated over time due to an attack, it can affect and degrade the performance of FPS protocol.

PEDAMACS

PEDAMACS is a TDMA-based MAC protocol used in sensor networks. It assumes that the base station is attached to a reliable and infinite power supply and is able to reach all the nodes in the network. The protocol consists of three phases [79]:

- Topology discovery
- Topology collection
- Scheduling

The network uses three different ranges for communication, where the longest range is used by the base station to reach all the nodes in one hop. The second range is used to determine the interference with other nodes, and the last level is for a node to reach its one hop neighbors. Topology discovery is done using Carrier Sense Multiple Access (CSMA). In the topology collection phase, the base station gathers information about the nodes in the network and builds a tree of the shortest paths. Next, the base station creates a transmission

schedule based on its knowledge of the network topology and broadcasts it to all the nodes. This schedule is structured such that nodes which do not interfere with each other during transmission may transmit in the same time slot. The base station periodically broadcasts the current time to all the nodes in the network to facilitate this scheduling. A guard interval on the time slots is used to compensate for clock drift between updates to the current time [79]. A compromised node might spoof the base station and broadcast an alternate time to its neighbors or it might attempt to jam the network to prevent the time updates from being received. If the induced difference in time between two nodes that are trying to transmit on adjacent slots is greater than the guard interval, packet loss may occur due to contention in the channel.

Since the base station can, by definition, communicate directly with every other node without routing packets through intermediary nodes, the base station can simply use an authenticated broadcast protocol to prevent a compromised node from injecting false packets. We see no defense to a jamming attack.

Estimation

To illustrate the effects of corrupted time synchronization on estimating state based on sensor readings from a sensor network, we give a simple example using the Kalman filter. The Kalman filter estimates the state of a discrete-time controlled process that is governed by a linear stochastic difference equation [80]:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \qquad x \in \Re^n \tag{4.5}$$

given the measurement $z_k \in \Re^m$, where

$$z_k = Hx_k + v_k \tag{4.6}$$

where $u(.)$ is the input to the system. The random variables w and v represent process and

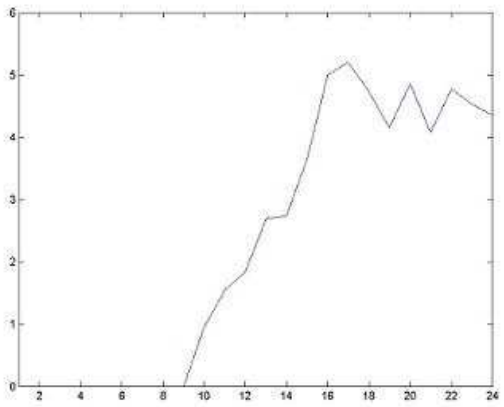


Figure 4.4. The y axis shows the norm of the difference between the results from the Kalman filter before and after de-synchronization. The x axis is the time of the corresponding observation.

measurement noise and are assumed to be independent random variables with normal distribution,

$$p(w) \sim N(0, Q) \qquad p(v) \sim N(0, R)$$

The Kalman filter estimates the state at every time step. We simulated the movement of an object using Equation 4.6, where the state is position and velocity of the object in two dimensions. We then used the Kalman filter to estimate the position and velocity of the object before and after modifying the time of some of the position observations, as might occur in an attack on the time synchronization in the sensor network. The norm of the error is shown in Figure 4.4. The de-synchronization is at time 10. From the figure, we can clearly see that the norm of the error grows as time goes on, meaning the estimate of the location of the object is far from its true location. If the purpose of the sensor network is tracking unknown objects passing through an area, the network administrator cannot pinpoint the location of the object.

Authenticated Broadcast(μTesla)

μTESLA is an authenticated broadcast protocol for sensor networks that forms the basis for some of countermeasures proposed here. It relies on an asymmetric mechanism to allow

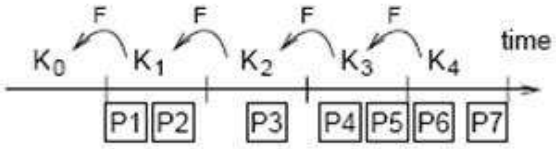


Figure 4.5. Example of $\mu TESLA$ [4]

nodes receiving a broadcasted message to verify the authenticity of the message. That mechanism is based on the arrival time of the messages, so one of the requirements of μTESLA is that the base station and the nodes have loose time synchronization.

Before beginning a broadcast that the nodes in the network may authenticate, the base station must bootstrap the protocol. Each message is transmitted in a time slot, although more than one message may be transmitted in one slot, and each time slot requires exactly one key. So the base station must transmit to each of the other nodes, authenticated with a private key shared only by the base station and that node, the length and starting time of the time slots. It also transmits a private key K_0. To generate K_0, the base station first generates a single key K_n and keeps it private. It then generates enough additional keys from K_n to transmit the packets in a sequence of time slots.

For example, if only one packet can be transmitted for each time slot, the base station would need $|S|$ additional keys. To generate a key from K_n, the base station applies a public one-way function F to K_n once for each of the required keys. That is, $K_i = F(K_i + 1)$. The sequence of keys constitutes a one way key chain. K_0 is last key in this sequence and is used to authenticate the messages in the first time slot, with K_i used for the ith interval and so on [4].

The nodes retain the packets they receive from the base station because they cannot authenticate them immediately. K_i is kept secret until after the *ith* interval. That is, at the $(i + k)$*th* interval, for some k transmitted during the bootstrap process, the base station reveals K_i.

With that key, the nodes can authenticate all the messages received at interval i (and all the messages in the previous time slots) [4].

To illustrate how the protocol works, consider Figure 4.5. P_1 through P_7 are the data packets, and K_0 through K_4 are the keys of the one way key chain. K_1 is used to decrypt P_1 and P_2, and K_3 is used to decrypt P_4 and P_5.

It is easy to see that μTESLA is vulnerable to time synchronization attacks. Although the protocol only requires loose clock synchronization, it does require that the network have a synchronized time. That is, the nodes must be able to unambiguously determine in which slot the messages occur. If a compromised node injects corrupted time updates that exceed the time synchronization error bound, the honest nodes would not be able to decide to which slot the messages or the keys belong. As a consequence, they would be unable to authenticate messages from the base station. It is interesting to note the symbiotic relationship between our countermeasures and μTESLA. On the one hand, our countermeasures rely on μTESLA or some other authenticated broadcast scheme. But without our countermeasures or some other scheme for securing time synchronization protocols, μTESLA is not secure. This is not to say that neither can function since they cannot function until the other has been established. Rather, if our countermeasures maintain time synchronization, they can continue to use μTESLA to do so. This is known as the boot strapping problem in the literature.

4.5 Countermeasures for Time Synchronization Attacks

We propose countermeasures for single hop networks and multi-hop networks separately. Each time synchronization protocol facilitates single hop synchronization by having a node periodically broadcast its local version of the global time and allowing the other nodes to synchronize their clocks to that time. Since multihop time synchronization algorithms are all extensions to that basic building block, our proposals for multihop networks are a superset of our proposals for single hop networks.

4.5.1 Countermeasures for Single Hop Networks

RBS is intended to be used in single-hop networks, although there is a multi-hop extension of the protocol. TPSN and FTSP can also be run in a single-hop network, however, and reportedly perform better than RBS. Regardless of which of these protocols is used in a single-hop network, however, the scheme we present is applicable since in the single-hop scenario all three protocols behave similarly.

In single-hop networks, every node is within radio range of every other node. We consider separately single-hop networks in which all the nodes are within range of a base station, a locus of trust, and those networks in which there is no such base station. The former is the case in many small deployments or deployments that include a second tier of supernodes. Such supernodes may maintain their time using GPS or by communication with another network that has a reliable time synchronization service. For these networks, the central challenge for time synchronization security is to prevent any node from spoofing the root node and sending erroneous updates to the global time.

Such a network might employ a network-wide symmetric private key to encrypt and authenticate messages from the root node, including time synchronization updates, to prevent spoofing of the root node and falsification of the time updates. There exist implementations of such a scheme for sensor networks, as mentioned above. An insider attack might occur, however, if a node were physically compromised. An adversary would gain access to the network-wide key and could falsify time synchronization updates. To detect the presence of falsified time updates, the nodes could easily look for redundant sequence numbers in the packets. Or, if the updates were expected to be sent at a regular, predefined period, the nodes could detect injected packets by a change greater in the frequency of the time-synchronization updates greater than would reasonably be expected shifting network conditions. If a node detects the presence of falsified packets, it could rely on a private key shared only with the base station to request and authenticate packets from the base station. However, such a scheme

would require each node in the network to have a private key with the base station, a highly inefficient proposition. Moreover, the radio traffic would increase linearly with the number of nodes requesting authenticated time updates. For these reasons, we recommend an authenticated broadcast scheme such as μTESLA as an effective means of maintaining time synchronization in single-hop networks with a base station. The base station should encrypt all time synchronization updates.

In single-hop networks that lack a base station or in which the base station fails, there is no trustable source of global time. In that situation, the nodes can elect a root node against which the other nodes can synchronize their clocks. FTSP provides one mechanism for electing a root node, as discussed above. Upon election, the new root node can bootstrap a new key chain for μTESLA using pair wise keys with the other nodes. However, we see no way to prevent a corrupted node from being elected as the root node under the FTSP scheme. Is fact, under the FTSP scheme, a corrupted node could become the root with certainty, as discussed in the attack section. A corrupted root node could send erroneous reference broadcasts that would cause the nodes to calculate an erroneous skew and offset. To prevent this, we propose that, instead of allowing a single node to be the source of time synchronization updates, a subset of the nodes act as the root on a rotating basis. In order to pick these subset of nodes, one way is to have the base station pick the node IDs on random and broadcast them (using encryption and authentication). Although this is a centralized solution, it will prevent problems associated with decentralized voting, such as ballot stuffing.

In addition, to prevent a corrupted node from impersonating other nodes, we recommend that all the nodes share a private key with that subset of the nodes in the single-hop network that may become a root. That is, if there are N nodes in the network and M $\ll$ N nodes may become a root node, there will be $N * M$ private keys in the entire network. Those keys could be used to bootstrap an authenticated broadcast scheme for each of the nodes in the rotation of root nodes. Given the update messages include a sequence number, a replay of the old messages from a valid root can not cause problems. Any corrupted nodes might continue to

send erroneous updates under this scheme, but the effects on the nodes' calculations of the skew and offset would be reduced. For instance, if there are C corrupted nodes in the network and every node has equal probability of being elected as a root node (a generous assumption, based on the previous discussion of the FTSP root-election scheme), then the probability that a corrupted node is elected as a root node is C/N. Suppose the nodes use linear regression with L data points to calculate the clock skew. Then without a root rotation scheme, with probability $\frac{C}{N}$, all the data points would come from a corrupted node. Given LS regression is not robust, it only takes one corrupted root node to completely corrupt the inferred data.
If we let M nodes be the root in a round robin fashion, then the probability of having at least one corrupted node being the root is:

$$p = 1 - \frac{\binom{C}{0}\binom{N-C}{M}}{\binom{N}{M}} \tag{4.7}$$

If we use LS to find the slope and intersection of the regression line using k data points, we have:

$$b = \frac{\sum_k (x_i - \overline{x})(y_i - \overline{y})}{\sum_k (x_i - \overline{x})^2} \tag{4.8}$$

$$a = \overline{y} - b\overline{x}$$

where $\overline{x}, \overline{y}$ are averages. Now if one data point is corrupted, i.e. the value of one of the x_i (and as a result y_i) is shifted by Δ. That will cause the averages to shift by $\frac{\Delta}{k}$. We call this shift in slope and y-axis intercept Δ_b and Δ_a. Therefore, if we have one compromised node introducing corrupted data in the network, we have a change of Δ_b in the clock skew. A compromised node is selected with probability p, so the expected value of the change in clock skew is $E(b) = p * \Delta_b$.

4.5.2 Countermeasures for Multi-Hop Networks

In multi-hop networks, at least one pair of nodes communicates with each other via a third node that routes messages between them. As in single-hop networks, there may or may not be a base station, a source of trustable time synchronization updates, although most deployments do have such a base station. However, some of the nodes in the network may not be in contact with the base station at all times. While there are various multi-hop schemes for time synchronization, as discussed above, they are all essentially schemes for repeating the synchronization scheme for single hop networks, estimation of a node's clock skew and offset by comparison with neighboring nodes, across larger networks. The difficulty in keeping each of the multi-hop schemes secure is in preventing corrupted nodes from increasing the synchronization error between the node with the reference clock and distant nodes. That is, the difficulty is in enabling distant nodes to verify the time synchronization updates they receive despite relying on nodes between the node with the reference clock and themselves to actively maintain–not simply route– those updates.

For accurate time synchronization, the nodes must receive the global time of their single-hop neighbors, not nodes that are more distant in the network. If a node computed the offset of its global time from the global time it received from a distant neighbor, that offset would include nondeterministic delays due to the time required to route the update message by intermediate nodes. Therefore, an authenticated broadcast scheme as provided by $\mu TESLA$ does not suffice to securely and accurately synchronize the time of the network to the broadcasting node. The root node could not maintain time synchronization by periodically flooding the network with authenticated clock updates.

Clock skew and offset could be calculated in the same way as for updates from a neighboring node. Such updates would, however, provide a useful approximation on the clock skew and offset for all the nodes in the network. Such updates might be sent an order of magnitude less frequently than the updates from a node's immediate neighbors. For each node, the error

in both the skew and offset derived from these updates would be primarily determined by the number of hops between the node and the base station, since there would be nondeterministic delays for each hop. In the absence of a base station, the long term trends in the skew calculated by a node might also serve as a useful approximation. So, our first proposal for multi-hop networks is to use such an approximation to get an upper bound on the error that can be induced by an adversary. If the node receives updates from its neighbors that yields a skew and offset sufficiently far from the approximated skew and offset of the root node or the long-term trends, it could ignore its neighbors and use that approximated skew and offset instead. However, this solution will not work if the corrupted node pulls the time away from the true time a little bit at a time, so that it is never too far off from the approximate bias and skew. Currently, we do not have any solution to this type of attack.

As discussed above, FTSP and TPSN rely on updates from a single neighbor node to calculate the offset and skew of its clock. One obvious means of increasing the reliability of these synchronization schemes, then, is to introduce redundancy into the system. This is our second proposal for multi-hop time synchronization protocols. In FTSP, it is especially easy to introduce redundancy. Rather than relying on a single update from a single node for each wave of updates from the nearest root node (i.e. for each seqNum), the nodes should record a subset S of the updates from their neighbors. This would increase the storage space required for the linear regression data points by a factor of S. In the current implementation of FTSP, the regression table holds 8 data points of 8 bytes each, 4 for the offset and 4 for the arrival time of that offset. If S were 5, for instance, this scheme would require accommodating 32 additional data points or $32 * (4 + 4) = 256$ bytes. Even on a mote class node, as described above, this is a reasonable additional memory requirement. Given this additional data, the nodes could take the median of the updates for any sequence number instead of whichever update is received first, which is the current scheme in FTSP. The nodes could initially have a set of private keys for their neighbors to authenticate these updates and prevent a compromised node from inserting false updates and corrupting that median. In addition, if a node

A saw a trend in which the updates from a neighboring node B tended to deviate from the other neighbors A by more than some threshold, node A could cease considering node B in its calculation of its clock skew and offset.

In that situation, where a node has become skeptical of the updates it is receiving from its neighbors, it may cease sending updates to its neighbors (FTSP) or children (TPSN). This is our third proposal for increasing the security of multi-hop time synchronization protocols, a policy of containment. Under this policy, corrupted nodes would be unable to de-synchronize nodes beyond their immediate neighborhood. However, this policy requires that nodes have multiple sources of time updates in case their source of updates ceases sending updates. In tree-based schemes, such as TPSN, that means having a mechanism whereby the children of nodes in the neighborhood of a corrupted node can find another parent outside the neighborhood of any corrupted node. If the tree for routing time synchronization updates is distinct from the tree for routing any other rout-ing tree used in the network, the tree already satisfies this requirement. However, maintaining multiple trees has the cost of redundant state at each node. In flooding-based schemes such as FTSP, this requirement is satisfied for free since each node receives updates from multiple neighbors and the loss of updates from a single neighbor can be ignored.

In multi-hop networks, if the base station fails or a node becomes disconnected from it, a new root must be elected against which nodes can synchronize their clocks. The same problems arise as discussed above in single hop networks-there is a risk that the newly elected root node will be a corrupted node. To avoid that situation, we recommend the same root rotation scheme as discussed above for single-hop networks.

Our final proposal for improving the security of multi-hop time synchronization is to make the LS linear regression used by each node to calculate the skew of its clock more robust. We propose using an algorithm similar to RANSAC [81], which would fit a model to a set of points that contain outliers in the following steps:

1. Set N=1, go to the next step.

2. Randomly select a subset of the data points of size m and build the initial model from these points

3. Determine the set of data points that are within ϵ of the model and call this set M. This set defines the inliers of the original data set.

4. If $|M|$ is greater than a threshold T, we need to re-estimate the model using all the points in M, and the algorithm terminates completely.

5. If $|M|$ is less than T, select a new subset and repeat from the second step on.

6. N=N+1, go to step 1.

7. Select the largest M from the iterative process above, and the model is re-estimated using all the data points in M.

In order to determine how many sample points m we need in each subset, we can use the following formula, where p is the probability that at least one of the subsets does not contain an outlier (usually taken to be 0.99), ϵ is the acceptable proportion of outliers, and N is the number of sub-sets:

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^m)} \tag{4.9}$$

We explain p and ϵ in our sensor network scenario. p is the probability that the attack is not successful since if a subset does not contain outliers, the result of estimation is going to be unaffected by the attack. This is the parameter that the network user has to set based on the application at hand. If the application can not tolerate large errors in estimation, then p has to be set high, otherwise, if the application can tolerate some level of error, p can be set to a lower value. ϵ is the fraction of sensors that are malicious. This is not a parameter that the network user can set, but it is possible to use an estimated value of it. In our case, the outliers are generated by an adversary (or a node with an especially erratic clock).

Before proceedings, we are going to perform a security analysis of the RANSAC method in terms of upper bounding the attacker's success probability as a function of number of

compromised nodes. The attacker's success probability is $1-p$, which we denote by s_A. By simplifying 4.9, we have:

$$s_A = (1-(1-\epsilon)^m)^N \tag{4.10}$$

For a fixed fraction of compromised nodes (ϵ), $(1-(1-\epsilon)^m)$ is an increasing function in m. For a fixed ϵ and m (number of data points in a random subset), $(1-(1-\epsilon)^m)^N$ is a decreasing function of N (the number of RANSAC iterations) and goes to zero as $N \to \infty$. An example of the behavior of this function, as m and N change, is shown in the plot of Figure 4.6. From this analysis we see that for a fixed m, $(1-(1-\epsilon)^m)^N$ is upper bounded by $(1-(1-\epsilon)^m))$. Now we let $\epsilon = \frac{k}{n}$ where k is the number of compromised nodes and n is the total number of nodes in the sensor network. We also note that the maximum value of $m = n$, meaning each subset consists of the data from all the nodes. Therefore, $(1-(1-\epsilon)^m)) = (1-(1-\frac{k}{n})^m)) \leq (1-(1-\frac{k}{n})^n))$. The last inequality follows from the fact that $(1-(1-\frac{k}{n})^m))$ is increasing in m. Now to find the upper bound, we let $n \to \infty$ and find the limit of $(1-(1-\frac{k}{n})^n))$. From the standard calculus, we have that:

$$lim_{n\to\infty}(1-\frac{k}{n})^n = \exp^{-k} \tag{4.11}$$

Therefore, $s_A = 1-\exp^{-k}$ which gives an upper bound on the probability of success as a function of total number of compromised nodes.

An alternative to both RANSAC and LS for linear regression that is more robust to outliers is LMS. However, LMS is a complex method and it may not be possible to implement it on a mote-class node. We have no such concerns about RANSAC, since it involves only a moderate modification of the basic LS scheme.In the LMS method, we still find the residuals, $r_i = Y_i - \hat{Y}_i$, as in the LS method, but instead of minimizing the sum of the squared of the residues we do the following:

$$\min med_i r_i^2$$

It is well known that LMS is more robust in the presence of outliers, so it is especially

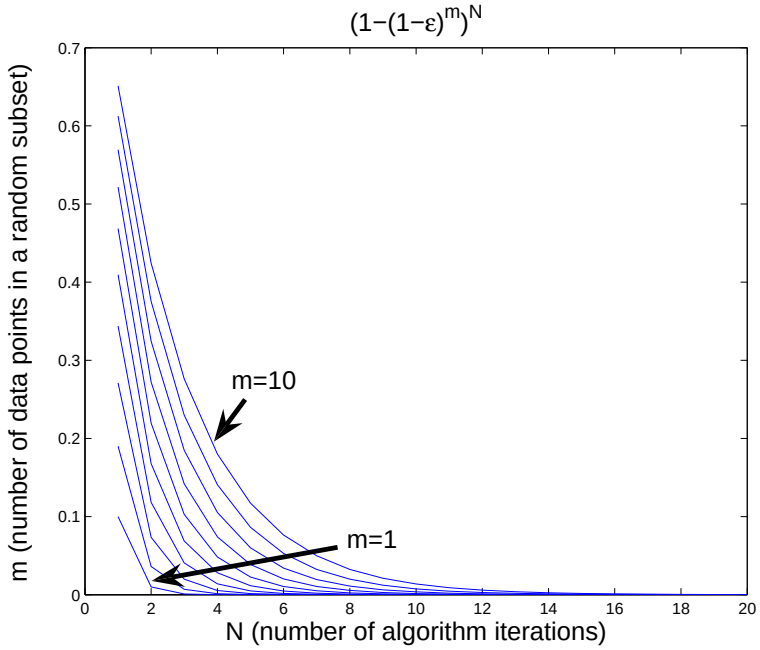


Figure 4.6. This plot shows how $(1-(1-\epsilon)^m)^N$ behaves as m and N increase.

appropriate for this application of regression where data points from corrupted nodes appear as outliers.

Finding the least median squares is a challenging optimization problem since we have to optimize the following equation:

$$\min_{b_0,b_1} med_i r_i = median\{(Y_1 - (b_0 + b_1 X_1))^2, ..., (Y_n - (b_0 + b_1 X_n))^2\}$$

There are software packages that can perform this optimization, but it is not clear whether it is possible to port the algorithms to a mote-class node.

4.6 Implementation of Attacks and Countermeasures for FTSP

In this section, we discuss the testbed implementation of attacks and countermeasures for FTSP. We explain which of the attacks described in the previous section were successful in desynchronizing the sensor nodes, and which countermeasures worked in reality.

4.6.1 Attack Implementation Results

As mentioned before, in FTSP the root node sends a time synchronization update once every TIME_SYNCRATE seconds. As a result, other nodes in the network will receive these update messages at the same rate of 1 message per TIME_SYNCRATE seconds. A possible attack scenario is that the compromised node would send time synchronization updates more frequently than TIME_SYNCRATE so as to increase the possibility of affecting the time of its neighboring nodes. At the same time, the compromised node must increase the sequence number accordingly to convince its neighbors to consider the compromised node's update message in their regression table (only the highest *seqNum* is considered in the FTSP regression table). Otherwise, as Figure 4.7 shows, the compromised node (B) will not have any affect on the time synchronization of its neighbors. The nodes in the neighborhood of A and B will receive node A's update before B, and since these updates have a higher sequence number, node B's updates are ignored. We implemented this attack on the sensor network testbed, and the result of the attack was in fact successful. Due to the similarity of the plot to Figure 4.9, we omit the plot of the results of TIME_SYNCRATE attack.

In FTSP only the root node is supposed to increase the value of the *seqNum* field of the time synchronization updates. This procedure is to facilitate the dynamic topology change for FTSP. A nice by-product of the *seqNum* is that it could potentially block some of the incorrect time synchronization updates propagated by compromised nodes. Therefore, the next attack

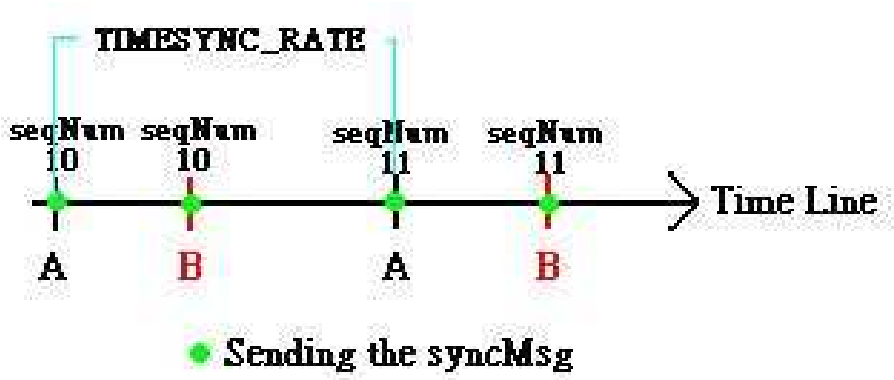


Figure 4.7. B is the compromised node and node A is a good node. Both nodes A and B have the same TIME_SYNCRATE, so if node A sends its time updates first, node B's updates will not affect the time synchronization of other nodes due to the *seqNum*.

scenario we tried on the testbed was to have the compromised node simultaneously falsify the *sendingTime* and the *seqNum*. This attack scenario was very successful and resulted in crashing the root node and FTSP completely. The testbed we used, shown in Figure 4.8, has 25 TelosB motes.

In this attack scenario, we programmed the compromised node to add a 'BAD_SEQNUM' to the *seqNum* of its time synchronization updates. In addition, the compromised node altered the value of *sendingTime* field when sending the time updates to its neighbors. From time $t_0 = 0$ to $t_1 = 180$ seconds, the motes were going through the process of root selection and initial data gathering (8 data points) for performing the synchronization. At time $t_2 = 600$ seconds, the compromised node was added to the experiment. The effect of this attack can be clearly seen from Figure 4.9. The attack took effect at $t_3 = 630$ seconds, as shown by the blue points which present the global time estimated by the nodes.

4.6.2 Countermeasure Implementation Results

In this section we describe the results of implementing some of the proposed countermeasures on the sensor network testbed. The first countermeasure implemented on the testbed

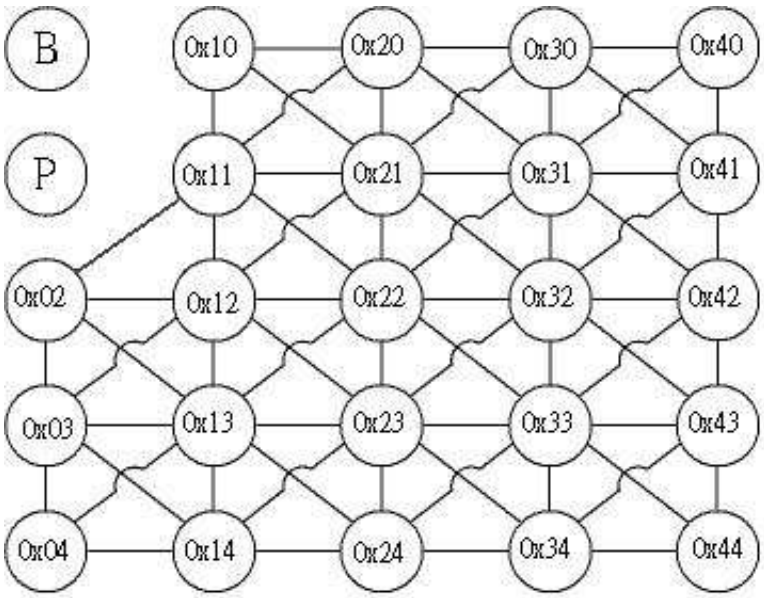


Figure 4.8. The testbed used in the experiments.

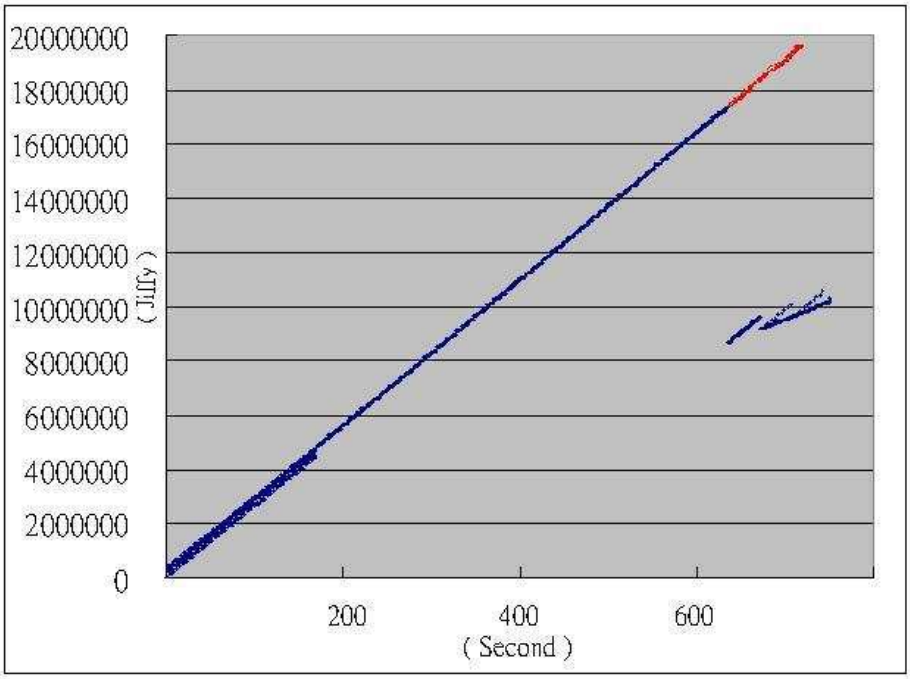


Figure 4.9. Result of seqNum attack. The blue line is the result of the actual regression on message updates, and the red line shows what the regression line should have looked like if there was no attack.

was to filter out bad data by collecting more data points from all the neighbors instead of receiving the synchronization message from only one node, and then filtering extreme values. To do this, we expanded the data table from 8 entries to 32 entries. The synchronization messages were collected from all neighboring nodes regardless of the message *seqNum*. As a result, the functionality of the *seqNum* field of FTSP was completely ignored.

The result of this experiment was surprising and revealed a feature of FTSP which is a side effect of having sequence numbers for each new update message. Although it is claimed in [82] that FTSP does not have any level of hierarchy, in reality it turns out that there is an implicit hierarchy due to *seqNum*, the local connection and the topology of the network. To explain this point, assume that the root node R broadcasts an update message M_t at some global time t. The nodes which are closest (within the communication range) to R will receive this update first and form the first level of hierarchy. This is due to the fact that in FTSP each node is only allowed to accept the updates with the highest (i.e. most recent) sequence number. We call this set of nodes L_1. These nodes perform the regression step, update their global time accordingly and broadcast their time update messages. The next set of nodes that receives this broadcast is the one-hop neighbors of L_1 nodes, which we call L_2. Continuing in this fashion, it is clear that after i broadcasts of the time update message, there are i sets of nodes (L_i), forming the hierarchy. Nodes in set L_i are one level higher in the hierarchy than nodes in L_{i+1}.

Now when the *seqNum* is not used, this hierarchy breaks down. In addition, the linear relation assumed between the global time and the local time of the nodes, in FTSP, is dependent on the fact that the updates are done in a short period of time where this linearity assumption holds true. However, when we add many data points from multiple neighboring nodes without using the sequence number of each data point in the regression table, the linearity assumption does not hold true anymore. This is due to the fact that the assumption of linearity of clock skew and offset in FTSP is based on a pair of nodes clock data and not a group of nodes clock data. The clocks of every pair of nodes have a linear relation, but when

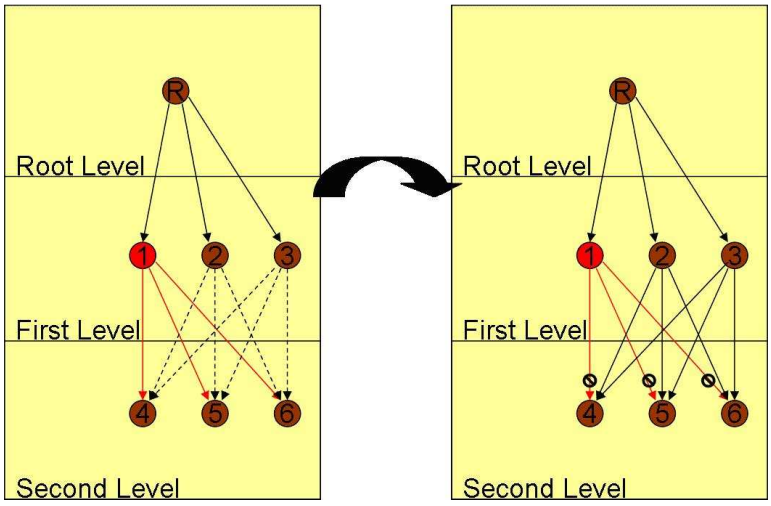


Figure 4.10. New data structure could block incorrect seqNum.

we combine the clock data from multiple nodes, the linearity does not hold anymore. This results in a very unstable linear regression, meaning the result of regression from one time update to the next fluctuates considerably. Figure 4.11 shows an example where regression is done on three sets of data, and as can be seen, each line has a different slope which results in an unstable regression. If we use all of the data, the regression line will try to fit all of the data, which will not yield a good estimate. This instability results in a significantly less precise time synchronization. Therefore, although our first countermeasure did not aid in preventing attacks, it revealed an important aspect of FTSP not known before, and that is the existence of implicit hierarchy in FTSP and the need for the sequence number as a mean of preserving the linear regression relation and stability of the algorithm.

Securing seqNum Attack: Given the result of our previous experiment, we need to use a countermeasure that will secure the sequence number functionality. Therefore, we created a new data structure, shown in Figure 4.10, containing three important parts of the data: the node ID for the incoming synchronization message (IncomingID), *seqNum* corresponding to each IncomingID, and a time out value for the corresponding node ID, called TIME_OUT. The TinyOS code for this structure is,

```
structure{
```

```
uint16_t nodeID;
uint8_t seqNum;
uint8_t timeOut;
uint8_t state;
}
```

Using this extended data structure, each node randomly chooses k nodes from its total of n neighbors and records these data, instead of recording only one neighbor's data points as in the original FTSP. Then, from among the randomly chosen neighbors, the node chooses the neighbor with the smallest *seqNum* and runs linear regression on the updates received from this node. This is different from the original FTSP where nodes choose the message with the highest sequence number from among 'all' the neighboring nodes. We call this countermeasure *seqNum filter*.

Given we run regression on the set of data points coming from the same node, the linear assumption will hold and the regression is stable. In addition, we keep the *seqNum* to preserve the implicit hierarchy in FTSP. The TIME_OUT feature is used to support the dynamic topology change that FTSP offers. For example, if one of the k neighbors of a node does not broadcast a time update for a period of TIME_OUT seconds, then the node can remove this neighbor from its regression table, and add a new neighbor in its place, which is randomly chosen from among the remaining available neighbors of the nodes. The results of applying the seqNum filter to FTSP is shown in Figures **??**. The node ID of the compromised node is 34, shown in Figure 4.8, and it starts sending wrong time updates at $t = 1600$ seconds (Figure 4.6.2). As seen from the plot, the attack clearly did not succeed in falsifying the global time estimation.

An explanation regarding the number of compromised nodes is in order. If node A has X compromised nodes and N good nodes as its neighbor, then we did not consider the case where $N = 0$, because if node A has no good node as its neighbor, there is no way for node

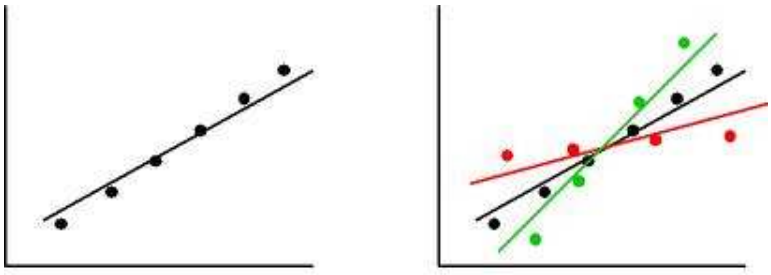


Figure 4.11. In the left diagram, the linear relation holds if the synchronization message is received form a single node. In the right diagram, where there are more nodes, the regression results lead to inconsistent data from different nodes, and the linear regression would be very unstable.

A to have the correct time. Therefore, we only considered the case when $N > 0$ and $X > 0$. Assume the table size of seqNum filter is K, then we have two cases,

- $X < K$: It means that the seqNum filter contains at least one good node with correct data. Therefore, node A can work correctly and the seqNum attack cannot succeed.
- $X \geq K$: seqNum filter randomly chooses K nodes from X+N. The probability of choosing at least one good node from X+N is $1 - (\frac{C(X,K)}{C(N+X,k)})$, which is the probability that the seqNum filter can defend successfully. It should be noted that the sequence number can not be zero, so a malicious node can not claim that its update has a seqNum of zero to make sure its data is chosen.

The *seqNum* filter proved to be effective in mitigating the effect of attacks on the sequence number and *sendingTime*. Using a combination of random neighbor selection and the seqNum helps make FTSP more robust to insider attacks while preserving the original features of FTSP, such as dynamic topology support.

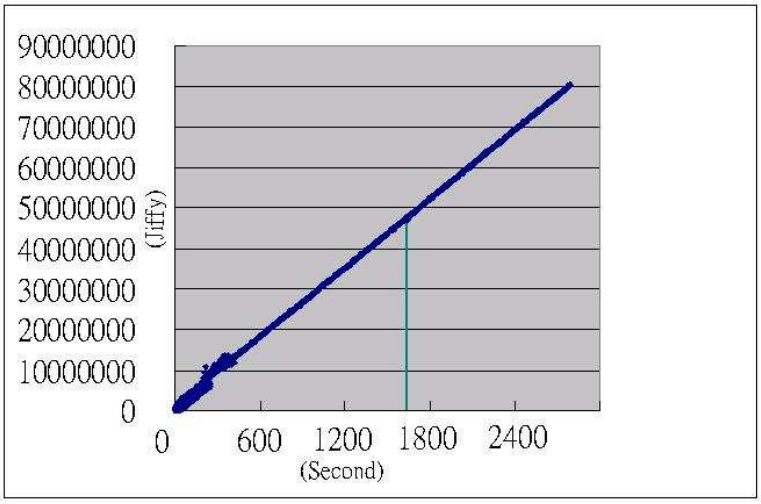


Figure 4.12. The seqNum filter used in combination with original FTSP. The attack started at t=1600 second but was unsuccessful. There were 4 Compromised Nodes in this experiments.

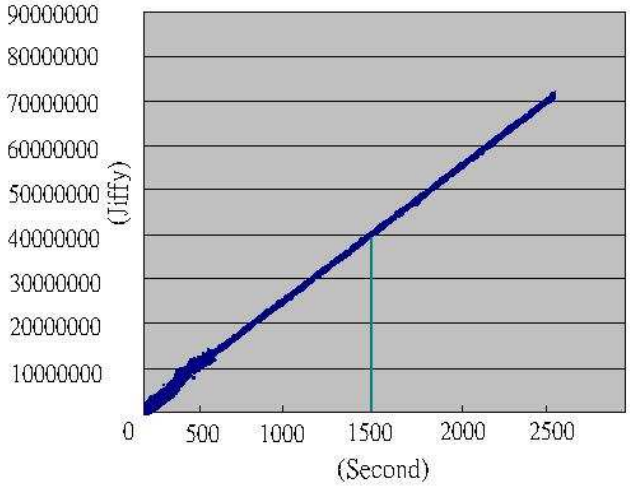


Figure 4.13. The seqNum filter used in combination with original FTSP. The attack was started at t=1500 with 4 compromised nodes, but it was unsuccessful.

4.7 Discussion

One of the fundamental tasks in sensor networks is the problem of time synchronization. Given the unattended nature of sensor networks, physical capture and compromise of the nodes is possible. Therefore, the attacker gains access to the network cryptographic keys and can participate in the legitimate communication among nodes. Therefore, designing a secure time synchronization protocol is crucial to maintaining the functionality of the sensor networks. In this chapter, we discussed various attacks on three major classes of time synchronization protocols. Moreover, we explained the effect of these attacks on upper layer applications relying on tight time synchronization. Finally, some of the described attack scenarios for FTSP were carried out on a testbed of 25 TelosB motes. We explained the degree to which each attack was successful in desynchronizing the network. Finally, some of our proposed countermeasures were implemented on the testbed to validate their usefulness in mitigating security attacks. We showed that adding the seqNum filter to the original FTSP helps mitigate the effect of insider attacks on this protocol.

Chapter 5

Multiple Object Tracking

One of the fundamental applications of sensor networks is performing multiple object tracking. In fact, the multi-target tracking is not a new problem inherent to sensor networks. It has been a subject of interest for many years due to its application in military and civilian areas. Examples include, ballistic missile defense, air defense, air traffic control, ocean surveillance [83]. In general, the most difficult aspect of multi-object tracking involves the problem of associating the observed sensor values with the appropriate object tracks[1][2].

In this chapter, we focus on the issue of robustness in the multitarget tracking in sensor networks. Most of the tracking algorithms developed for sensor networks, and in general networks, assume that the data gathered from the sensors is not faulty and is only tainted by an ambient Gaussian noise. However, this assumption might not hold true. The sensor data can be corrupted due to various causes, spanning from the result of an explicit attack to a fault in the sensing unit hardware which is a likely scenario. An explicit attack in our context refers to the possibility of a node compromise. Once a sensor node has been compromised, it could be reprogrammed to report faulty or malicious observations.

[1] Permission to use parts of these works for the purposes of this dissertation have been granted by the IEEE and/or the appropriate copyright holder(s).

[2] Part of the contents of this chapter are based on [84].

From this point on, we collectively refer to malicious and faulty sensor observations as *anomalies*. We consider anomalies different from other forms of interferences, such as noise, clutters and jamming. Interferences are admitted in an operating network, and research has been developed to cope with this issue. Also tracking systems based on sensors outputs take into consideration the false alarms due to clutters, noise, and undetected targets. The signal to noise ratio (SNR) measure the ability of the system to overcome unwanted signals. These can be internal if it is noise inherently generated by the electronic device or external if the noise is generated for example by the background scene surrounding the target. Also, clutters have an external source. They are in fact any unwanted signal which are of no interest to the sensor network user.

Anomalies due to sensor malfunctioning, failure, or malicious attacks are of different nature. Currently, multitarget tracking algorithms do not take into account these possibilities. However, these anomalies could impact the data association of the tracking algorithm and result in formation of incorrect object tracks. The concept of anomalies in the context of sensor networks and tracking is intimately connected to the problem of formulating secure and robust hypotheses by the tracking algorithm.

In this chapter, we focus on two problems related to robust tracking: 1) Robust hypothesis formation , and 2) Robust local aggregation. The goal of all the object tracking algorithms is to associate the time-stamped sensor observations with a moving object. This is done by either forming hypothesis about the object tracks and choosing the strongest hypothesis, or through maximizing the posterior probability of a track given the observations. Either of these methods rely on the assumption that the underlying time-stamped sensor observations are not faulty, meaning the sensors are reporting the true value they sense, and the only source of corruption is the White (Gaussian) noise. However, this assumption might not hold true if some of the sensors have hardware failure or are compromised by an attacker. To alleviate this problem, we propose a robust version of the multiple hypothesis tracking algorithm in Section 5.2.

Given the limited communication and computation power in sensor networks, many algorithms, including some tracking algorithms, employ a local aggregation phase (Section 2.8) to decrease the amount of data that needs to be communicated. Therefore, it is of great importance to ensure that the local aggregation process is robust to attacks, and mainly to insider attacks (node compromise). Figure 5.1 shows an example of how the tracking algorithm can be thrown off by faulty observations. We propose a reputation-based framework in Section 5.3 to tackle this issue.

The rest of the chapter is organized as follows: Section 5.1 gives some background information on the problem of multi object tracking. Section 5.2 discusses the basics of multiple hypothesis tracking, the sensor fault model, and the extended hypothesis formation which includes the possibility of faulty sensors. Section 5.3 gives the background on reputation systems, develops the reputation system at the local level for the tracking application, and gives the simulation results of including the reputation system in the tracking algorithm.

5.1 Background Work

In general the tracking algorithms work as follows: when an object moves in the vicinity of a proximity sensor, such as a Passive Infrared (PID), the sensor records a signal strength. The standard analytical model for the recorded signal strength by sensor node s_i is [5],

$$z_i = \begin{cases} \frac{\beta}{1+\gamma\|s_i-x\|^\alpha} + w_i, & \text{if } \| s_i - x \| \leq R_s \\ w_i, & \text{if } \| s_i - x \| > R_s, \end{cases} \tag{5.1}$$

where α, β, and γ are internal parameters specific to the sensor hardware. The signal strength z_i is normalized so that w_i has the standard Gaussian distribution. If z_i is below a given threshold, namely η, the sensor can not differentiate the signal reading from noise, and therefore, the collected signal is not used in the data fusion phase of the tracking algorithm.

These observations (signal strengths) are gathered by a central base station and combined

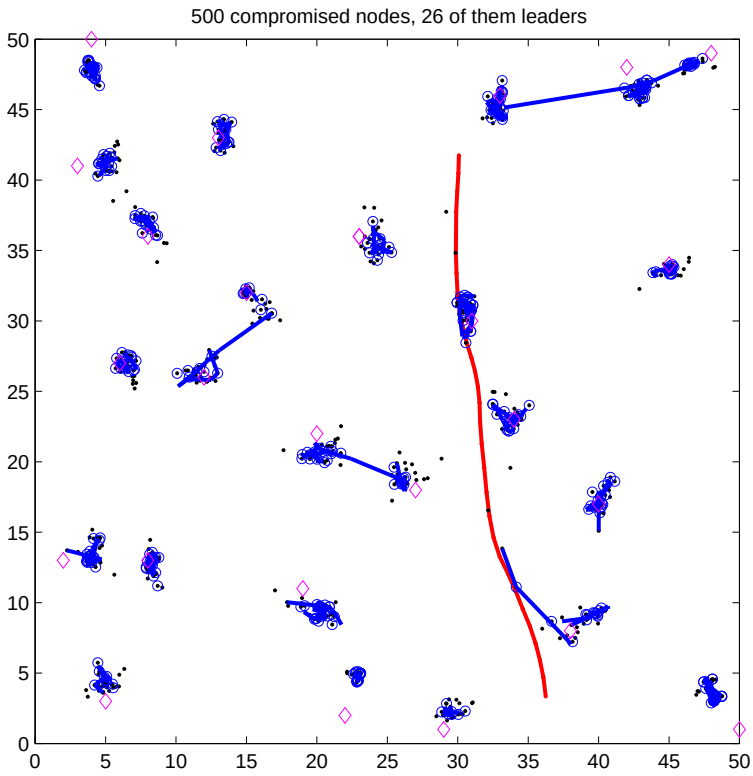


Figure 5.1. The blue lines are the tracks formed at the base station. The red line is the actual track of the moving object.

to form object tracks using a data association algorithm. Several methods for data association have been proposed in the literature and different methods are often discussed in the estimation and tracking literature [85]–[87]. In general, multi target tracking deals with state estimation of an unknown number of targets. Some methods are special cases which assume that the number of targets is constant or known. The observations are considered to originate from targets if detected or from clutter. The clutter is a special model for the so-called false alarms, whose statistical properties are different from the targets. In some applications only one measurement is assumed from each target object, and in other applications several measurements are available [88]. A general example of observation association with tracks is shown in Figure 5.2 (courtesy of [5]).

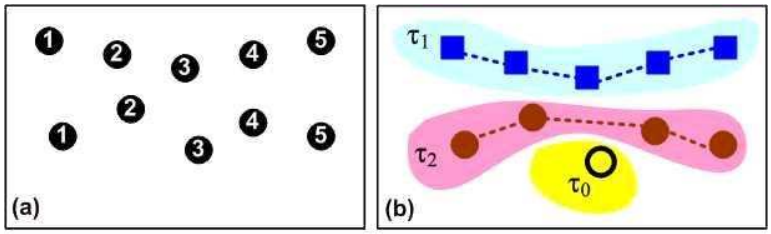


Figure 5.2. Each circle is a sensor observation and the number within each circle denotes the time of the corresponding observation. b) shows a possible association of observations to two tracks and one false alarm [5].

In this chapter we deal with two main approaches to multitarget tracking. The first is Multi-Hypotheses Tracking (MHT) developed by Reid [83]. The second approach is the work in [5] where the authors design a Monte Carlo Markov Chain Data Association (MCMCDA) algorithm. We use MHT in formulating our robust hypothesis tracking problem, and MCMCDA in making the local aggregation process more robust.

MCMCDA is a hierarchical algorithm that consists of a data fusion component at the node level and a data association component at the base station level. MCMCDA is capable of tracking multiple objects without knowing the number of objects a priori. The number of objects is estimated along with estimating the tracks. Once all the observations have

been received from each of the sensor nodes that have readings, the data association phase starts. In this phase, the goal is to divide the observations from the sensor nodes into non-overlapping partitions, such that each observation belongs to only one partition. One of the partitions holds the false alarm observations. In order to find the most likely partitioning of the observed data, Y, the posterior of the track partition, ω, is maximized, i.e.,

$$P(\omega|Y) \propto \prod_{t=1}^{T} p_z^{z_t}(1-p_z)^{c_t} p_d^{d_t}(1-p_d)^{u_t} \lambda_b^{a_t} \lambda_f^{f_t} P(Y|\omega)$$

where z_t is the number of objects terminated at time t, a_t is the number of new objects at time t, d_t is the number of detections, f_t is the probability of false alarms, λ_f is the false alarm rate, λ_b is the birth rate of a new object, p_z is the probability of an object disappearing, and p_d is the probability of detection.

This is accomplished by applying Markov Chain Monte Carlo (MCMC) to associate the observed data to form tracks. The method in [**?**], [5] uses a sliding window of size w_s, so as to avoid explosion of the data space. At each time step, the Markov Chain is initialized with the estimated track from the previous step. One of the components of MCMC is choosing a proposal distribution. The authors in [5] consider five possibilities for the proposal distribution: 1)birth/death 2) split/merge 3) extension/reduction 4)track update and 5) track switch. At every step of the MCMC, the proposal distribution is randomly chosen from a probability distribution which is an input parameter to the MCMCDA algorithm.

In MHT the idea is to formulate hypotheses by associating new observations to an existing pool of rated hypotheses in all the possible ways and then calculate the new rating recursively. To avoid redundancy, we postpone the complete treatment of the MHT algorithm to Section 5.2. It is worth mentioning that MHT algorithm has been used in other applications beside object tracking. For example, MHT is the building block of the Process Query System (PQS) Engine [89]. PQS is a powerful software front-end to a data base or a real time sensing infrastructure, that allows users to define processes at a high level of abstraction and submit process definition as queries. PQS has proven to be very successful in many challenging ap-

plications, such as, detection of cyber attacks, detection of attacks on cognitive radios, active worm detection for network monitoring, visual tracking, plume detection and tracking, social networks analysis [90]–[94].

In both of these approaches, the probability density of the state of a single target given sequence of all its observations is performed through Kalman filtering (although this can be generalized to other models). The major differences between the MCMCDA and MHT can be summarized as follows. The MCMCDA hypothesis generation is based on both current and past observations whereas the MHT algorithm uses recursive filtering techniques to estimate the current state of each track as a function of the current observation being assigned to it and the previous estimate. As a result, MCMCDA does not need to maintain multiple hypotheses, but it needs to reconsider all the observations in order to compute a new hypothesis. MHT updates the estimate of each track much more efficiently, but it needs to consider several possible associations of observations to existing tracks (hypotheses). This leads to a potentially geometric growth of the hypotheses set. We can say that the time complexity of MCMCDA corresponds to the space complexity of MHT. In MCMCDA the length of the random walk through the hypothesis space affects the accuracy of the estimation process while in MHT the accuracy depends on the ability to maintain all the relevant hypotheses of consistent tracks that can be formed in memory. Therefore, MCMCDA performs considerably better when the application at hand has to process the data online and does not have a large amount of memory and computation power.

In this section, we focus on the MHT algorithms and expand its hypothesis space by including the probability of faulty or malicious sensor behavior when associating observations with tracks. The next section describes our analytical framework in detail.

5.2 Multiple Hypothesis Tracking

In this section, we describe out approach to extending the hypothesis formation of the MHT algorithm to make it more robust in the presence of anomalous sensor observations.

5.2.1 Detection Model

Let the area of surveillance be **A** which is covered by a total of n sensor nodes. We define a new entity called 'macro sensor' which is a cluster of neighboring sensor nodes that have a collective probability of detection p_d. Each macro sensor consists of $m_{cluster}$ actual physical sensor nodes. The probability p_d depends on the fraction of the $m_{cluster}$ sensors that are functioning properly.

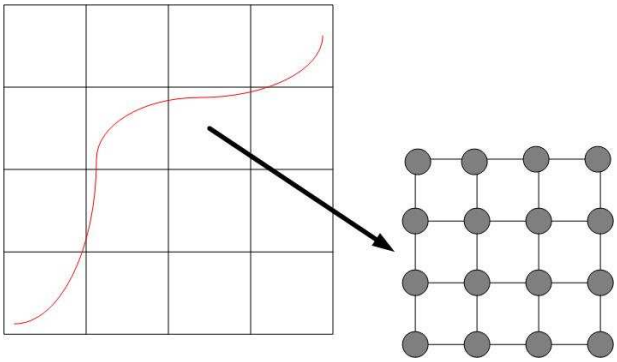


Figure 5.3. The red line is the object moving through the sensor network. Each macro sensor, shown on the left grid, consists of a number of actual sensors as the blown up grid on the right hand side of the figure shows.

5.2.2 State Space Model and Kalman Filter

Each moving target is represented by a state vector $x \in \mathbf{R}^d$ where d is the dimension of the state variables. The discrete-time dynamics of the state vector evolve according to,

$$x(k+1) = \Phi x(k) + \Gamma w(k) \tag{5.2}$$

where,

Φ = The state transition matrix

Γ = The disturbance matrix

w = White noise sequence of Gaussian random variables

with zero mean and covariance of Q

The equation for the measurements z is,

$$z(k) = Hx(k) + \upsilon(k) \tag{5.3}$$

where

H = Measurement matrix

υ = White noise sequence of Gaussian random variable

with zero mean and covariance R.

If the measurements can be uniquely associated with each target, then the conditional probability distribution of the state variables of each target is multivariate Gaussian distribution given by the Kalman filter [80]. The Kalman filter estimates the state of a discrete-time controlled process that governed by a linear stochastic difference equation. The mean $\bar{x}$ and covariance $\bar{P}$ of this distribution evolve according to the *time update* equations,

$$\begin{aligned} \bar{x}(K=1) &= \Phi\hat{x}(k) \\ \bar{P}(K+1) &= \Phi\hat{P}(k)\Phi^T + \Gamma Q\Gamma^T \end{aligned} \tag{5.4}$$

The conditional mean $\hat{x}$ and covariance $\hat{P}$ evolve according to the *measurement update* equations,

$$\begin{aligned} \hat{x} &= \bar{x}(k) + K[x(k) - H\bar{x}(k)] \\ \hat{P}(k) &= \bar{P} - \bar{P}H^T(H\bar{P}H^T + R)^{-1}H\bar{P} \\ K &= \hat{P}H^TR^{-1} \end{aligned} \tag{5.5}$$

5.2.3 Algorithmic Details

In this section, we describe the MHT algorithm as presented in [83]. The basic idea is to generate a set of data association hypotheses that account for all the possible origins of every single measurement received from sensors. These hypotheses are measurement oriented, meaning every possible target is listed for every measurement.

Let the measurements in the data set k be denoted by $Z(k) = \{Z_m(k), m = 1, 2, \ldots, M_k\}$. The total set of measurements up to the data set k is denoted by $Z^k = \{Z(1), Z(2), \ldots, Z(k)\}$. The set of all hypotheses at the time of data set k which associate the set Z^k with targets or clutter is $\Omega^k = \{\Omega_i^k, i = 1, 2, \ldots, I_k\}$. $\bar{\Omega}^m$ is the set of hypotheses after the mth measurement of a given data set has been processed by the algorithm [83].

Once a new set of measurements $Z(k+1)$ is received, the algorithm forms a new set of hypotheses Ω^{k+1} as follows: $\bar{\Omega}^0$ is initialized by setting it equal to the set of hypotheses from the previous step, i.e. Ω^k. The new set of hypotheses, Ω^m is formed in a repetitive fashion from Ω^{m-1} and every measurement in $Z_m(k+1)$. The hypotheses in this new set are interpreted as the joint hypotheses that $\bar{\Omega}_i^{m-1}$ is true and that measurement $Z_m(k+1)$ was emitted from target j. If $j = 0$, then the measurement is a false alarm. If j has any other value besides 0, then it is either the value from a previous known target, or it is one greater than the current number of tentative targets. After the completion of this repetitive process, the algorithm arrives at a new set of hypotheses denoted by $\bar{\Omega}^{M_{k+1}}$, which will be used as the initial set of hypotheses for the next step of the algorithm when new observations from sensors come in.

Another important factor of the MHT algorithm is to determine the probability of each hypothesis, i.e. how likely each new formed hypothesis is. To find the analytical formula for this probability, let P_i^k denote the probability of hypothesis Ω_i^k. Then, we have,

$$P_i^k = P(\Omega_i^k | Z^k) \tag{5.6}$$

Now let ψ_h be the association hypothesis for the current data set, meaning, it denotes the hypothetical assignment of each measurement in the data set $Z(k)$ with a target. Using Bayes formula, we get,

$$P(\Omega_g^{k-1}, \psi_h | Z(k)) = \frac{1}{c} P(Z(k) | \Omega_g^{k-1}, \psi_h) P(\psi_h | \Omega_g^{k-1}) P(\Omega_g^{k-1}) \tag{5.7}$$

where the factor c is a normalizing factor. The first term on the right hand side is the likelihood of the measurement $Z(k)$ given the association hypothesis,

$$P(Z(k) | \Omega_g^{k-1}, \psi_h) = \prod_{m=1}^{M_k} f(m) \tag{5.8}$$

$f(m) = \frac{1}{V}$ if the mth measurement is from a clutter or a new target (V is the volume of the region of the sensor deployment), and $f(m) = N(Z_m - H\bar{x}, B)$ if the measurement is from a confirmed target or a tentative target whose existence is implied by Ω_g^{k-1}. The second term in Equation 5.7 is the probability of the current data association hypothesis given the prior hypothesis, and includes a number of useful information [83],

- The number of measurements associated with the prior targets $N_{DT}(h)$, the number of measurements associated with false targets $N_{FT}(h)$, and the number of measurements associated with new targets $N_{NT}(h)$.
- Configuration of those measurements which are from previously known targets, those which are from false targets, and the measurements that are from new targets.
- The source of each measurement that has been assigned to be from some previously known target.

In addition, the hypothesis Ω_g^{k-1} encapsulates the information for the number of previously known targets $N_{TGT}(g)$, which includes both the number of tentative targets whose

existence is implied by the prior hypothesis and the number of confirmed targets. However, based on ψ_h only N_{DT} of these targets are detected by the sensor.

It is assumed that the number of previously known targets that are detected follows a binomial distribution, and the the number of false targets has a Poisson distribution. Moreover, the number of new targets follows a Poisson distribution [83]. Given these assumption, we arrive at,

$$P(N_{DT}, N_{FT}, NNT|\Omega_g^{k-1}) =$$

$$\binom{N_{TGT}}{N_{DT}} P_D^{N_{DT}}(1-P_D)^{(N_{TGT}-N_{DT})} \tag{5.9}$$

$$\times F_{N_{DT}}(\beta_{FT}V)F_{N_{NT}}(\beta_{NT}V)$$

where we have the following definition for each parameter in the above formula,

$P_D =$ Probability of detection

$\beta_{FT} =$ Density of the false targets

$\beta_{NT} =$ Density of previously unknown targets which have been detected

$F_n(\lambda) =$ Poisson distribution for n events with rate λ

$M_k = N_{DT} + N_{FT} + N_{NT}$

where M_k is the total number of measurements

The next step is to find the number of possible configurations for assigning N_{DT}, N_{FT}, and N_{NT}, so we have,

$$\binom{M_k}{N_{DT}}\binom{M_k - N_{DT}}{N_{FT}}\binom{M-k-N_{DT}-N_{FT}}{N_{NT}} \tag{5.10}$$

Therefore,

$$P(Configuration|N_{DT}, N_{FT}, N_{NT}) = \frac{1}{\binom{M_k}{N_{DT}}\binom{M_k - N_{DT}}{N_{FT}}\binom{M-k-N_{DT}-N_{FT}}{N_{NT}}} \tag{5.11}$$

Once we have the number of configurations, we have to find the number of ways the N_{DT} measurements can be assigned to N_{TGT} targets for each given configuration. This value can be found from $\frac{N_{TGT}!}{(N_{TGT}-N_{DT})!}$, so,

$$P(assignment|configuration) = \frac{N_{TGT}!}{(N_{TGT} - N_{DT})!} \tag{5.12}$$

Putting all of the equations together and simplifying gives us the key equation for the probability of each hypothesis,

$$P_i^i = \tfrac{1}{c} P_D^{N_{DT}} (1 - P_D)^(N_{TGT} - N_{DT}) \beta_{FT}^{N_{FT}} \beta_{NT}^{N_{NT}} \times [\textstyle\prod_{m=1}^{N_{DT}} N(Z_m - H\bar{x}, B)] P_g^{k-1} \tag{5.13}$$

This equation is used for the hypothesis generation process to find the probability of each data association hypothesis in an interactive fashion. In the next section, we modify this equation to incorporate additional parameters for anomalous observations.

It should be noted that by unfolding this equation and adding the birth and death events for targets that appear and disappear from the sensed area one obtains the posterior density used in the MCMCDA algorithm.

5.2.4 Extended MHT Algorithm

In this section, we present the extended MHT framework which shows how to incorporate anomalies into the original MHT algorithm. First, we define our fault model and then present

a new definition of the hypothesis space that accounts for observations generated by faulty sensors.

5.2.5 Fault Model

We first need to define our notion of anomaly more precisely. We consider a proximity sensor S_i in a sensor network that issues an observation z_i according to Equation 5.1, repeated here for convenience,

$$z_i = \begin{cases} \frac{\beta}{1+\gamma\|s_i-x\|^\alpha} + w_i, & \text{if } \| s_i - x \| \leq R_s \ , \\ w_i, & \text{if } \| s_i - x \| > R_s \ . \end{cases}$$

In a realistic scenario, we should expect the sensor to detect, miss a detection or report a false target with probabilities p_1, q_1 and r_1 respectively. In this point of view, detections, misdetections and false reports are considered part of the ordinary behavior of operating devices and are factored in the way hypotheses are built and rated in the MHT method.

Now we introduce a new factor that we call 'anomaly'. The term anomaly refers to the permanent condition of a sensing device that results from an unrecoverable damage or failure. Once a sensor is broken, or compromised, it exhibits an anomalous behavior which consists of detecting, miss a detection or reporting false targets with different probabilities p_2, q_2 and r_2 respectively. In addition, we assume that an individual sensor has a status that changes according to a stochastic process (see next Section for details).

Since the condition of being anomalous is an 'unrecoverable' state of a sensor hypothesis, valid tracks will have to satisfy a consistency property. What we mean by unrecoverable is that when a sensor node is compromised or has hardware failures, it is not possible for it to go back to a sate of being 'good' unless the failure is fixed, or the compromised node is taken out of the network. Therefore, if a sensor is assumed to break at time t, observations reported by it at successive instants of time must be treated accordingly, meaning they all have to be considered faulty and can not be treated as 'good' data.

In our enriched model, each sensor $s \in S$ has a status $S(t) \in \{o, b, f\}$ (operating, just broken, failed). The difference between b and f is that a sensor breaks at time step t, and once it is broken, at any time step after that it has a status of f. Moreover, each observation Z is issued by a specific sensor.

Table 5.1. Parameters used in the extended MHT framework

DT: detected targets	o: good sensor
FT: false detections	f: faulty sensor
NT: new targets	m: faulty but not broken sensors

5.2.6 Extended Posterior Density

We process observations that come from individual sensors. Sensors have an operating status that affects the probability that the measurement is valid or a false echo (clutter). Our idea is to record the status of sensors during the hypothesis formation phase and to weight the probability of each hypothesis accordingly.

In this section we intend to generalize formula 5.13 that refers to processing a single measurement from what is called type II sensors. Let us focus on a specific sensor s. We define the following stochastic process $S(t_k)$ about the status of s at time t_k:

$$S(t_k) = \begin{cases} o & s \text{ is operating at time } t_k \\ f & s \text{ failed at time } t' \leq t_{k-1} \\ b & s \text{ was operating at time } t' \leq t_{k-1} \\ & \text{but broke at time } t \in (t_{k-1}, t_k] \end{cases}$$

Moreover, each sensor produces two kinds of measurements: $K =$ "valid measurement" of targets already detected or of new targets, and $\overline{K} =$ "invalid measurement", which are false echos and clutter. The probability of event K depends in our model on the status of the sensor that has issued the measurement.

Let N_{TGT} and β_{NT}, as in [83], be respectively the number of previously detected targets and the average density of new targets in the sensed area V.

Following Reid's approach we write the probability of an hypothesis using Bayes theorem as

$$P(\Omega_g^{k-1}; \psi_h | Z^k) \sim P(Z(k) | \Omega_g^{k-1}; \psi_h) \cdot P(\psi_h | \Omega_g^{k-1}) \cdot P(\Omega_g^{k-1})$$

where the dependency on past data Z^{k-1} has been dropped for brevity.

Now, we are processing directly one measurement at a time only, i.e. $|Z(k)| = 1$ and the association hypothesis ψ_h describes the following situation: $Z(k)$ is either a valid or invalid measurement. In case it is valid then it is either a new target or a previously detected target, otherwise it is a false echo.

Assume that the possible association hypotheses are numbered from 0 to $N_{TGT} + 1$. ψ_0 defines $Z(k)$ as a clutter; ψ_i, with $1 \leq i \leq N_{TGT}$ associates $Z(k)$ with track i from Ω_g^{k-1}. $\psi_{N_{TGT}+1}$ defines $Z(k)$ as a new target. Then the likelihood of the measurement given the association hypothesis is as in [83]:

$P(Z(k) | \Omega_g^{k-1}; \psi_i) \sim$

$$\begin{cases} N(Z(k) - Hx_i, B_i) & 1 \leq i \leq N_{TGT} \\ \frac{1}{V} & i = 0 \\ \frac{1}{V} & i = N_{TGT} + 1 \end{cases}$$

In fact, in the case $1 \leq i \leq N_{TGT}$, the density of $Z(k)$ is Gaussian with mean Hx_i and covariance matrix B_i, as guaranteed by the Kalman filter associated with track i; in the case $i = 0$ (clutter) and $i = N_{TGT}$ (new target), the density of $Z(k)$ is assumed to be uniform over the sensed area V. The probability of the association hypothesis becomes

$$P(\psi_i|\Omega_g^{k-1}) \sim$$

$$\begin{cases} P(K; S(t)|\Omega_g^{k-1}) \frac{N_{TGT}}{\beta_{NT}V+N_{TGT}} & 1 \leq i \leq N_{TGT}, \\ P(\overline{K}; S(t)|\Omega_g^{k-1}) & i = 0, \\ P(K; S(t)|\Omega_g^{k-1}) \frac{\beta_{NT}V}{\beta_{NT}V+N_{TGT}} & i = N_{TGT}+1. \end{cases}$$

This second factor includes the status of the sensor. In particular, in case $1 \leq i \leq N_{TGT}$ the probability that $Z(k)$ is to be associated with an existing track i is proportional to the probability that the measurement itself is valid and that it belongs to an existing target. In case $i = 0$ the probability that $Z(k)$ is to be associated with a clutter is equal to the probability that the measurement is not valid. Finally, in case $i = N_{TGT} + 1$ the probability that $Z(k)$ is to be associated with a new target is proportional to the probability that the measurement itself is valid and that it does not belong to an existing target.

This approach can be implemented very easily. It is necessary to estimate $P(K|S(t))P(S(t)|\Omega_g^{k-1})$. This can be accomplished by employing the negative exponential distribution. Moreover, each time a sensor breaks its new status needs to be recorded in the new hypothesis being formed.

Notice that the process for a sensor to fail permanently is irreversible and can be described by the Probabilistic Deterministic Finite State Automaton (PDFA) in Figure 5.4. This means that the status of a sensor transitions irreversibly from status o to status b and then to status f. Consequently, the following is true:

$$\begin{aligned} P(S(t_k) = o \mid S(t) = o, t < t_k) &= \alpha(t_k)\,, \\ P(S(t_k) = b \mid S(t) = o, t < t_k) &= 1 - \alpha(t_k)\,, \\ P(S(t_k) = o \mid S(t) \neq o, t < t_k) &= 0\,, \\ P(S(t_k) = b \mid S(t) \neq o, t < t_k) &= 0\,, \\ P(S(t_k) = f \mid S(t) = b, t < t_k) &= 1\,. \end{aligned}$$

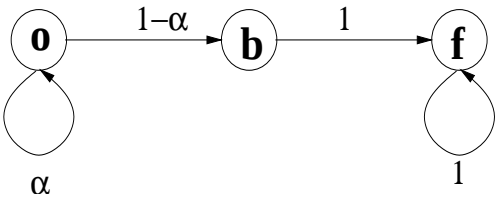


Figure 5.4. PDFA describing the transitions of sensors' status

Our MHT-based approach will typically produce an intractable number of hypotheses, which will limit the efficaciousness of our approach as a simple object tracking system. However our objective is to design effective diagnostic tests for sensor networks. One way to assess the functionality of the sensors is to circulate a known number of test targets with an accurate model of their kinematics and process the best hypotheses calculated by the tracking system. This is similar to training the system.

The number and trajectories of the test targets should be planned so as to maximize coverage of the deployment area. The processing of the hypotheses should then allow the estimation of the number and identity of the faulty or compromised sensors. This approach is particularly valuable in situations where it is impossible to verify the functionality of each sensor by direct interaction or when the sensor has been compromised. It is a very difficult task to determine when the sensor is being malicious. Our approach is essentially based on quantifying the deviation of the sensor response from the expected position of the target up to a reasonable statistical fluctuations caused by the white noise.

5.3 Reputation-Based Framework for Object Tracking

In this section, we focus on the problem of robust local data aggregation for hierarchical tracking in sensor networks, and develop a reputation system to aid mitigate the effects of contaminated sensor observations. We take the hierarchical multi-object tracking algorithm

that was proposed by authors in [5] as the baseline algorithm and we develop our reputation system for this tracking algorithm. The algorithm in [5] relies heavily on the local aggregation process as the first phase of its tracking algorithm. We develop a reputation-based framework for robust local data aggregation.

The concept of reputation has been used in many fields such as, economy, sociology, and computer science. A reputation system consists of two parts: 1) an algorithm or mechanism by which the entities in a system rank one another based on their opinions (observations of actions), 2) a way by which these rankings are propagated from one time step to the next. The ranking can be a number on a chosen scale, such as a one to ten scale. However, it is generally easier to normalize these rankings to arrive at a value in the interval $[0, 1]$ because the ranking can then be translated into 'the probability of collaboration in the future given the past actions of an entity'.

Reputation systems have proven useful as a self-policing mechanism to address the threat of compromised entities. They operate by identifying selfish nodes, and isolating them from the network. It should be noted that the reputation systems are not the definite answer to the problem of misbehavior. The reason is that an intelligent attacker can behave properly for a period of time to build up a good reputation, then misbehave but only to the extent that does not throw him outside of the 'trusted' entities. By doing so, the attacker can cause damage to the system while remaining on the 'good list'. Despite this shortcoming of the reputation systems, they do prove useful in detecting some level of misbehavior by malicious entities.

Centralized reputations systems were popularized by the internet [95]–[97]. An example of this system is Ebay's rating system [96]. Decentralized methods were then created for use in ad hoc networks [40]. The *CORE* reputation system [41] and the CONFIDANT protocol [42] use a watchdog module at each node to monitor the forwarding rate of the neighbors of the node. If the node does not forward the message, its reputation is decreases, and this information is propagated throughout the network. Each node also uses the second hand

information from other nodes to find the overall reputation of a node. Over time the bad behaving nodes are less trusted and will not be used in forming reliable paths for routing purposes. These two protocols differ in how they use the second hand information, how to punish bad behavior and how to instill trust for the node which misbehave temporarily. A formal model for trust in dynamic networks is introduced by Carbone et. al [43]. The most relevant work to ours is presented in [44]. The authors suggest a high level reputation system frame work for sensor networks. The main difference between our work and [44] is that the authors only suggest a 'watchdog' mechanism to determine the reputation of each node. They state that there is no unifying way in which reputations can be assigned, i.e. the mechanism to assign reputation has to be context dependent and varies based on the application at hand. Our contribution is to develop the assignment mechanism for the specific case of multi-object tracking application in sensor networks.

5.3.1 Problem Statement

We consider the problem of robust data aggregation at the local level for multi-object tracking in sensor networks. We assume that the nodes are equipped with cryptographic keys for secure communication [15].

Threat model: Our threat model is as follows: the adversary has physically captured a subset of the nodes, and therefore has access to the network keys and can participate in communication with other nodes without being detected. The objective of the adversary is to use the compromised nodes to inject faulty data into the network, where the data refers to the signal strength that each sensor detects(observation of the sensors). We look at the effect of this type of attack on the multi-object tracking algorithm proposed in [5].

We consider 'decentralized trust' as opposed to 'centralized trust' [98], meaning there is no global entity that assigns the reputation values, and all the nodes contact this central unit to find the reputations of the other nodes. In contrast, the 'decentralized trust' refers to the

situation where each node is responsible for determining the reputations itself. In addition, we deal with 'reactive' reputation computation [98], meaning the nodes compute the reputation only when they become leaders, as will be explained in the later sections.

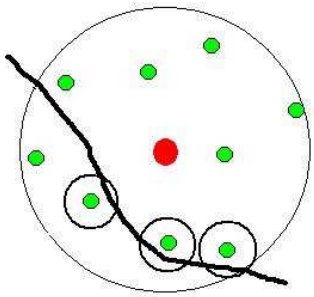


Figure 5.5. The red node in the middle of the figure is a supernode which can communicate with all the regular node within its communication range (the large circle). When an object passes through the sensor field, the regular nodes close to the track of the moving object pick up a signal. The sensing range of the regular nodes is shown with the small circles.

5.3.2 Hierarchical Multi-object Tracking

We base the design of our reputation system on the multi-object tracking application in [5]. Therefore, we briefly describe the basic operations of this multi-object tracking algorithm.

The sensor network is comprised of N_s nodes. A few of these nodes are called supernodes since they have more communication and computation power. We refer to the rest of the nodes as regular nodes. These sensors are distributed throughout the region $R \in R^2$ A single supernode, denoted by S_i, governs a given area $A_i \subset R$ and can communicate with all regular nodes in A_i as well as the neighboring supernodes. Regular nodes, denoted by s_{ij} can communicate with other regular nodes within a range of $2R_s$,where R_s is the sensing range of a node, Figure 5.5.

When a moving object crosses the sensor field, the sensor nodes that are close to the object track get triggered, i.e. each sensor node records a signal strength which is called the observation of that node. For example, if a node is equipped with passive infrared sensor and an object gets close to the node, the object blocks the infrared port, and as a result the node records a signal strength, or 'senses the object'.

The observations from different nodes are combined to form object tracks using a distributed tracking algorithm. The way the tracking algorithm works is to keep a state variable

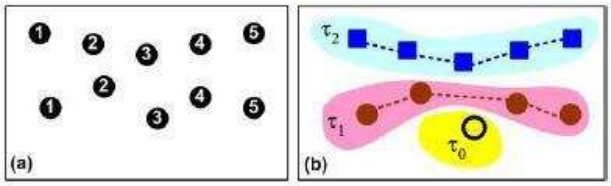


Figure 5.6. Partitioning the observations of the sensors over time into object tracks. This figure is taken from [5].

for each object it is tracking. The track for the object is formed by finding the most likely path given the observations[3]. The tracking algorithm developed in [5] is a hierarchical algorithm that consists of a data fusion component at the node level(local) and a data association component at the supernode level(global). This algorithm is capable of tracking multiple objects without knowing the number of objects a priori. The number of objects is estimated along with estimating the tracks.

Our reputation mechanism is specifically designed to deal with the local data fusion. We describe these two steps of the tracking algorithm [5] in more detail in the following subsections. From this point on, we refer to regular nodes simply as nodes.

[3]This is accomplished using a Bayesian inference framework and maximizing the posterior probability estimation of the track given the observations. The general approach to finding the posterior probability estimation is using Markov Chain Monte Carlo (MCMC).

Data Fusion

The data fusion component takes care of aggregating each of the observations by nodes in a local area into a single, fused observation that can be used by the data association component at the supernode level. We denote the location of the i-th node by s_i and the recorded signal strength by z_i. The sensor model used is:

$$z_i = \begin{cases} \frac{\beta}{1+\gamma\|s_i-x\|^\alpha} + w_i, & \text{if } \| s_i - x \| \leq R_s \\ w_i, & \text{if } \| s_i - x \| > R_s, \end{cases} \tag{5.14}$$

where α, β, and γ are internal parameters specific to the sensor hardware, and x is the position of the object with respect to a given Cartesian coordinate system. The signal strength z_i is normalized so that w_i has the standard Gaussian distribution, $w_i \sim N(0, 1)$. If z_i is below a given threshold, namely η, the sensor can not differentiate the reading from noise, and therefore, the reading is not used in the data fusion.

Each sensor looks at the incoming signal strength readings from its neighbors. If the number of incoming readings is below some threshold, the sensor does not have enough neighbors to validate its own reading. Thus its reading will not be incorporated in the data fusion outcome. For example, in [5], the authors suggest that each node has at least 4 neighbors. If the sensor has enough incoming readings, it will compare its reading to that of its neighbors' to determine if it has the highest signal strength [5]. The sensor that has the highest signal strength, declares itself the leader and fuses its observation with those of its neighbors. Given that sensor s_{i0} is the leader, i.e. z_{i0} is larger than all incoming readings $z_{i1}, z_{i2}, ..., z_{ik}$, the position of the object is determined as follows:

$$\hat{s}_{obj} = \frac{\sum_{j=1}^{k} z_{ij} s_{ij}}{\sum_{j=1}^{k} z_{ij}}, \tag{5.15}$$

The logic behind this equation is as follows: the higher the signal strength recorded by a node, the closer the object is to that node. Therefore, if we take the weighted average of the position of all the nodes that have sensed a moving object, we will get an estimate of

the position of the object. The wights used are the signal strength observed by each node. This value is sent to the governing supernode. Each supernode collects the estimated object positions from the leaders in its governing region in order to perform the data association phase.

Data Association

Once the supernode has received the fused observations from each of the local areas in its region, data association is performed. The goal is to associate the fused data points to form a track, as shown in Figure 5.6. The details of the data association step are beyond the scope of this discussion, and we refer the reader to [5] for the details.

Each supernode maintains its own set of tracks through performing the data association process. Thus, a single object can have multiple tracks maintained by different supernodes. These multiple tracks from different supernodes are combined at the main supernode, i.e. the base station, using another iteration of the data association algorithm. The motivation for our work is the following: if the compromised nodes inject faulty observations into the network, the local fused data will become contaminated. Therefore, when the supernodes use these contaminated fused data, the tracks they form will be faulty. As a result, there is a need have a mechanism in place that is able to filter out the bad data, which will help the track formation process.

5.3.3 Reputation System

As authors in [44] have pointed out, the main challenge in designing a reputation system is to develop the mechanism by which the reputation is assigned for the particular application at hand. Our main contribution in this section is to develop a reputation assignment mechanism to specifically address the data fusion phase of the tracking algorithm.

When a malicious node injects faulty observations into the network, it can throw off the value of the fused data calculated through Equation 5.15. This will affect the accuracy of the tracks that are formed as well as the number of the estimated tracks. By assigning reputations to nodes over time, we are able to weight their observations by the corresponding reputation. This approach will result in minimizing the effect of faulty observations on the data fusion by suppressing the readings from the malicious nodes.

Our reputation assignment scheme attempts to give reputations to nodes at a local level and does not attempt to solve the problem of secure leader selection. As mentioned in 5.3.2, the node that claims to have the highest signal strength becomes the leader. In order to assure that a compromised node does not become the leader, we have to develop a secure leader election protocol, and have methods in place at the supernodes to determine the compromised nodes and filter out their data. Secure leader election is outside the scope of our work, but it is worth noting that there are standard distributed coin-flipping algorithms in the literature, using cryptographic commitments.

In our scheme, every time a node becomes the leader, it updates the reputation of its own neighbors, and keeps a table of the reputations values. There is no sharing of the reputation tables among nodes, i.e. no use of the second hand information. At each time step, the nodes which have readings will receive an instantaneous reputation rating. For each node, the instantaneous rating is combined with its ratings from the previous time steps to form an overall reputation for the node. Observations from a node are then weighted by this overall reputation when data fusion is done.

Reputation Assignment Mechanism

During a single time step,t, the leader of a local neighborhood assigns reputations to those nodes it receives readings from. We call this the instantaneous reputation. To determine what

the reputation of each node is, the leader uses a method similar to RANSAC(Random Sample Consensus) [99].

RANSAC relies on random sampling selection to search for the best model to fit a set of points that is known to contain outliers. In effect, RANSAC can be considered to seek the best model that maximizes the number of inlier data. The following is the set of steps taken by the RANSAC algorithm to find the best model parameters:

1. Randomly select a subset of the data points of size m and build the initial model from these points.
2. Determine the set of data points that are within ϵ of the model and call this set M. This set defines the inliers of the original data set.
3. If $|M|$ is greater than a threshold T, we need to re-estimate the model using all the points in M, and the algorithm terminates.
4. If $|M|$ is less than T, select a new subset and repeat 2.
5. After N trials the largest M is selected, and the model is re-estimated using all the data points in M.

In order to use RANSAC, we need to determine the number of points in the small subsets, the number of iterations, and the threshold used to identify a point that fits well.

Following the RANSAC approach, our scheme chooses subsets of neighboring nodes at random. These subsets are then used to find the estimate of the object location using Equation 5.17. The size of each subset and the number of subsets required to get a good estimate can be found using the following formula [99]:

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}, \tag{5.16}$$

where N is the number of subsets, s is the number of samples in each subset, and ϵ is the percentage of contamination. p is the probability of having at least one subset free of outlier

data. This probability is usually set to 0.99, meaning with probability 0.99 there are no outliers in the data sample. The justification for the above equation can be found in [99]. In our problem, we need to ensure that at least half of the chosen subsets are free of outliers, so as to get a good overall estimate. This is due to the fact that the median is used as the best estimate. Therefore, we need to have $(1-\epsilon)^s < 0.5$, or $\epsilon \leq \frac{0.69}{s}$.
The reason for using the above scheme is to find the best possible estimate of the position of the object given the observed data. Since there is no model to fit the data to, as RANSAC suggests, we need to find a 'best guess' for the object location as a reference point in order to later assign the reputation to each node.

Equation 5.15 is used on each selected subset to find one value for the fused observation. We call this value S^i_{fused} to refer to the fused value for the i^{th} subset. The only difference in calculating S^i_{fused} is that we use the overall reputation for each node as the weighting factor in the summation, i.e.:

$$\hat{s}_{obj} = \frac{\sum_{j=1}^{k} z_{ij} s_{ij} l_{ij}^{t-1}}{\sum_{j=1}^{k} z_{ij} l_{ij}^{t-1}}, \tag{5.17}$$

where l_{ij} is the overall reputation of node j from neighbor i. Once the estimated object locations are determined from all the subsets, the median of those estimates is calculated. Namely we find the median of S^i_{fused}, where $i \in \{1, ..., N\}$, and N is calculated from Equation 5.16. We use the median as opposed to mean since it has been shown analytically [48], [100] that the median is a robust estimator with a *breakdown point* of 50% in contrast to the mean which has a breakdown point of zero. [4]

The median, m, and the corresponding subset of nodes that give the value m are assumed truthful. We call this subset S_{trust}. S_{trust} is used as a starting point for determining the reputations of the remaining nodes. If multiple subsets of nodes result in the same value of the median, we choose the subset of nodes with the lowest variance as the starting point. There

[4]The breakdown point of an estimator refers to how resistent the estimator is to the percentage of outliers and contaminated samples. The higher this point is, the more robust the estimator will be.

are two counters, $(\alpha_{ij}, \beta_{ij})$, which are updated for the instantaneous reputation. Positive reputation is kept by α_{ij} and negative reputation is kept by β_{ij}. The nodes in S_{trust} receive an instantaneous reputation of $(1, 0)$ since we assume they are truthful. The leader node, which will be passing the fused signal to the supernode, also receives a $(1, 0)$ reputation since the data it has will be used in tracking calculations.

To determine the reputation of the remaining nodes in the neighborhood, we pick one node,s_{ij}, at a time and add it to the subset S_{trust}. Then the weighted sum of this node along with the nodes in S_{trust} is found using 5.17. We call the result of this calculation $\hat{m}_{ij}$. This new estimate, $\hat{m}_{ij}$, is compared with the median m and the reputation for the node s_{ij} is assigned as follows:

$$(\alpha_{ij}, \beta_{ij}) = \begin{cases} (1,0), & \text{if } |\hat{m}_{ij} - m| = 0 \\ (\frac{T-|\hat{m}_{ij}-m|}{T}, 0), & \text{if } |\hat{m}_{ij} - m| \leq T \\ (0,1), & \text{if } |\hat{m}_{ij} - m| > T, \end{cases} \tag{5.18}$$

where T is a threshold to determine how far $\hat{m}_{ij}$ can be pulled away from the median m, while node s_{ij} still gets a positive rating. Note that when $|\hat{m}_{ij} - m| < T$, the node does not get a positive rating of 1. Instead, it gets a fraction proportional to how close it is to the median. The reason for this assignment is that a node that has a closer value to the median should be rated higher than a node that gives an estimate further away from the median. The threshold value T has to be adjusted so that it does not assign low rating to nodes that have a noisy observation within the reasonable bounds. At the same time, it can not be too large to allow the compromised nodes to inject faulty readings. The nodes whose estimate $\hat{m}_{ij}$ falls outside the threshold get a negative instantaneous reputation.

It is important to give a little analytical justification for the procedure described above and Equation 5.18. First, we need to define the concept of W-estimators. W-estimators are robust location estimators which are closely related to the well-known M-estimators [101]. Robust

estimators, such as W- and M-estimators, try to minimize some function of the data. However, the numerical values of some robust estimators can only be obtained after an iterative process due to lack of any closed forms for the solution to these estimators. Usually one has to choose a starting value Γ_0 and use the iterative process to arrive at new estimates, Γ^*. A W-estimator starts with Γ_0 equal to the median of the observations. Then the iteration process consists of only *one* step, and the final value of the estimate is given by,

$$\Gamma^* = \frac{\sum w(u_i).y_i}{\sum w(u_i)} \tag{5.19}$$

where y_i is the ith observation, $y_i - \Gamma_0$ is the residual, $u_i = \frac{y_i - \Gamma_0}{c\sigma}$ is the relative deviation, σ is the standard deviation of the residuals, c is some constant to be chosen, and $w(u)$ is a symmetric weighting function conventionally taken to give the value 1 at $u = 0$ and decreasing as $|u|$ gets larger [101].

Now we can map the W-estimator defined above to the reputation assignment Equation 5.18. In our framework, the starting value Γ_0 (median of the observations) is m (the median of the fused observations from the randomly chosen subsets). By looking at Equation 5.18 more carefully, we observe that $u_i = \frac{|\hat{m}_{ij} - m|}{T}$ and $w(u_i) = \frac{T - |\hat{m}_{ij} - m|}{T}$. By our definition, $w(u_i) = 1$ at $u_i = 0$, is symmetric, and is decreasing as $|u_i|$ becomes larger (up to the point that the deviation is outside the threshold region). Therefore, we can see that the value of the instantaneous positive reputation α_{ij} is essentially Γ^* (the new location estimation). Given W-estimators are robust estimators, our reputation assignment mechanism is robust to outliers. It also demonstrates that Equation **??** is based on concrete analytical foundation rather than an ad-hoc method.

5.3.4 Reputation Fusion

The instantaneous reputations for the nodes, calculated in the previous section, are aggregated over time to calculate cumulative positive and negative reputation ratings which we

represent by (r_{ij}^t, s_{ij}^t). Old reputation values may not be as relevant for the cumulative ratings, as a node may change behavior over time. Thus, to determine the cumulative ratings, we use a discounting factor, λ, that guarantees the old reputations will be gradually forgotten. The ratings are determined by the following equation:

$$\begin{aligned} r_{ij}^t &= \lambda r_{ij}^{t-1} + \alpha_{ij} \\ s_{ij}^t &= \lambda s_{ij}^{t-1} + \beta_{ij}, \end{aligned} \tag{5.20}$$

Using the cumulative ratings the overall reputation, l_{ij}, of a node at time t is calculated. The overall reputation varies in the range $[0, 1]$ where 0 represents the most untruthful and 1 represents the most truthful. In order to determine where a node's overall reputation falls on this scale, we use the Beta distribution. We consider the sequence of observations as a sample from a binomial distribution, i.e. a sequence of independent coin tosses, with a bias parameter *P*. To make the statement clear, the *head* corresponds to an honest node and the *tail* corresponds to a compromised node, and the bias is the overall rating of the node. We can then estimate the rating of a node using Bayesian parameter estimation of the binomial distribution. If we use a Beta distribution as the prior distribution, the formula for calculating the posterior parameter estimate actually coincides with the maximum likelihood estimate. The fact that the posterior probability of binary events is most accurately represented by the Beta distribution, has been shown mathematically in [102]. The Beta distribution is a two parameter distribution whose parameters are denoted by a and b. The parameter a measures the number of successes (r_{ij}^t) and b measures the number of failures (s_{ij}^t). The bias estimate of the underlying binomial distribution, *P*, is given by the mode (average) of the Beta distribution, i.e. $P = \frac{a-1}{a+b-2}$.

The Beta distribution is well suited for our purposes since at each time step a node is either truthful or is lying, which is a binary event. The overall reputation is modeled as the expected value of the Beta distribution [102]:

$$l_{ij}^t = \frac{r_{ij}^t - s_{ij}^t}{r_{ij}^t + s_{ij}^t + 2}, \tag{5.21}$$

This give us a reputation ranging from $[-1, 1]$; therefore, we need to rescale this interval to the interval $[0, 1]$. This overall reputation is used in the next time step as a weighting factor in (5.17) for calculating the median.

Two comments are in order regarding our reputation system. We are not considering second hand information, i.e. the information coming from the neighbors, since that will give the compromised nodes a window of opportunity to attack and degrade the reputation of their neighbors. The use of second hand information is useful if each node has a scheme for filtering out badmouthing. This in turn will add to the overhead incurred by the reputation system. Therefore, we chose not to add the second hand information in the reputation updating formula. Secondly, the reputations are stored locally at each leader node. Every time a node becomes the leader, it will update its reputation table. This scheme avoids having to broadcast the reputation tables of each node to its neighbors. We speculate that the local reputations will converge to the true reputation values over time. This is due to the fact that over time almost all the nodes will become leaders. The validation of this speculation is the focus of our future work.

5.3.5 Simulation Results

To provide empirical evidence on the performance of our reputation system, we performed a number of simulations. Again, in this work we do not consider secure leader election. Therefore, in the simulations we do not allow for the compromised nodes to claim to be leaders. In our settings, the surveillance region is a square grid of size $50mx50m$. There is one node placed at each corner of each square. Therefore, there is a total of 2500 nodes in the simulation grid. The number of objects we want to track is n_i. The sensing range of the sensors,R_s is set to $1.5m$. This sensing range has been shown to be optimal in terms of low estimation errors for different speeds of the moving targets [5]. We are simulating real sen-

sors in our experiments, therefore, the sensor readings are noisy, and the noise is represented by a Gaussian standard distribution with mean of zero and variance of one.

The metric used to quantify the performance of the multi-object tracking algorithm is the average error in the number of tracks estimated by the algorithm compared to the actual number of tracks, ϵ_K where [5]:

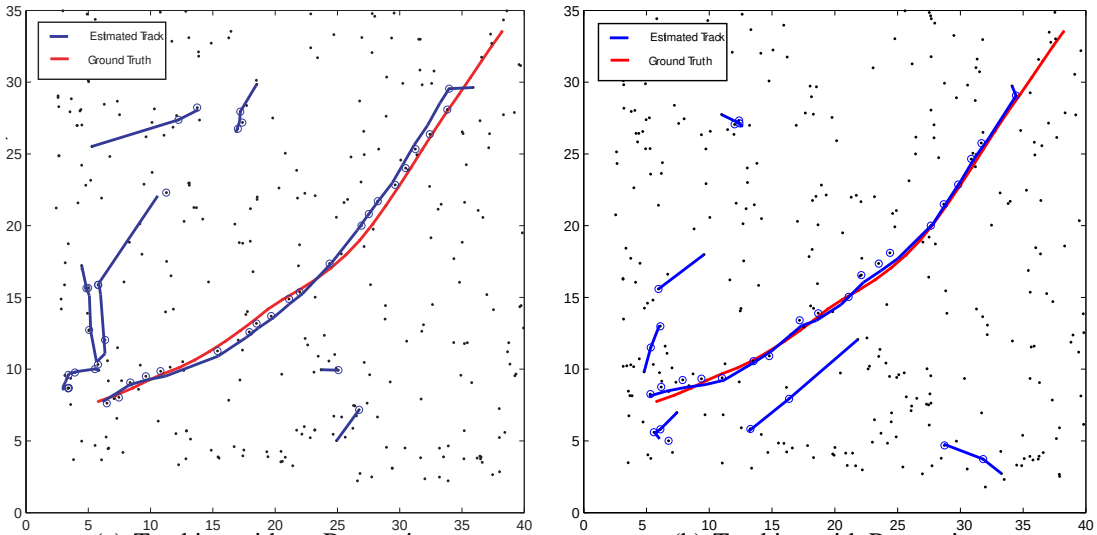
$$\epsilon_K = \frac{1}{W} \sum_{w=1}^{W} \mid K_w - \tilde{K_w} \mid$$

In our simulations we look at multiple scenarios. In the first scenario we keep the number of tracks,n_i, constant and change the number of compromised nodes from 250 to 1000. The results are shown in Figures 5.8 and 5.9 for differing values of n_i. In our second scenario, we fix the number of compromised nodes and vary the sensing radius,R_s, from 1.5m to 3m as shown in Figures 5.10 and 5.11. In these scenarios we use a threshold value, $T = 0.4$, and $s = 3$, where s is the number of nodes in each subset.

We can see from Figure 5.7 that the reputation framework gets rid of many of the spurious estimated tracks and decreases the size of the ones that remain. In addition, the estimated tracks associate with the ground truth are more accurate and more closely follow the actual movement of the object. In our simulations, we did not allow for the compromised nodes to become leaders. As we discussed in the previous section, filtering out the compromised leaders requires either a secure leader election process, or a centralized scheme at the supernodes to keep reputations for the leader nodes.

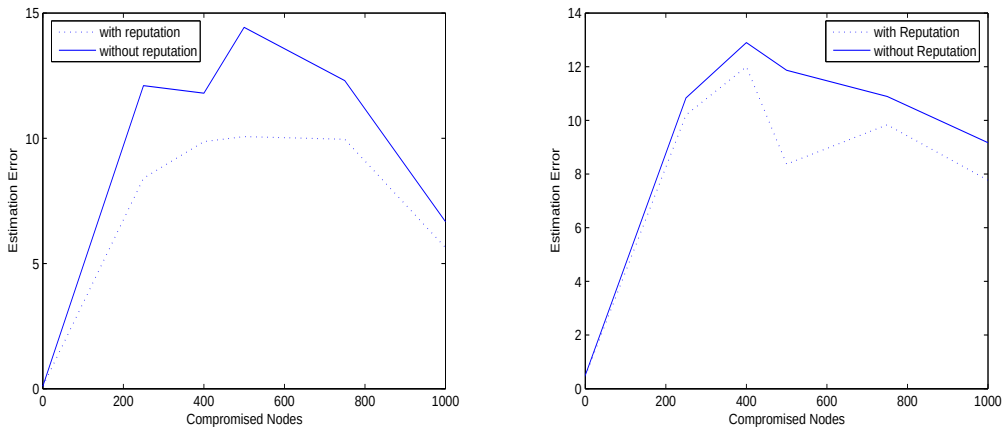
5.4 Discussion

In this chapter we looked at the problem of robust tracking for sensor networks, at both local and global level. We extended the multiple hypothesis tracking algorithm framework to explicitly take the possibility of anomalous sensor observation into account. Next, we developed a reputation system based on the ideas of robust statistical estimation to deal with



(a) Tracking without Reputation (b) Tracking with Reputation

Figure 5.7. Estimated object track compared to ground truth



(a) 3 Objects (b) 6 Objects

Figure 5.8. Estimation error ϵ_K

anomalous observations at the local level. The reputation system helps filter out the anomalous sensor observations before they are fed into the higher level tracking algorithm. This mitigates the effect of bad observations on the overall tracking performance.

As part of the future work, we need to run simulations on our extended MHT algorithm to verify its effectiveness as a function of: number of compromised to number of not compromised sensors ratio, number of test target to number of sensors ratio, measurement noise, and the kinematics noise.

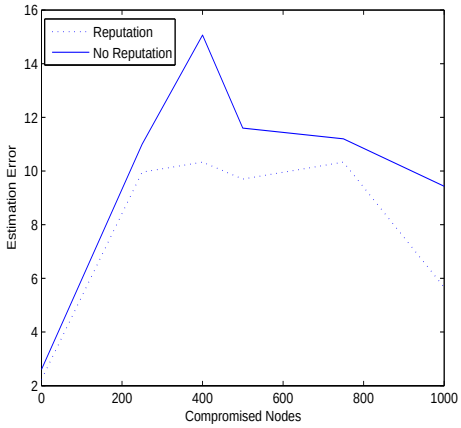
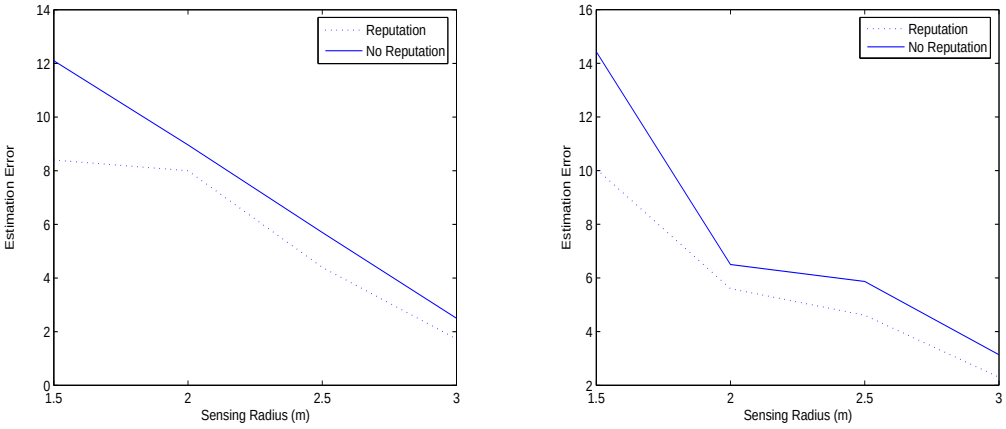


Figure 5.9. Estimation error ϵ_K.



(a) 250 Compromised Nodes (b) 500 Compromised Nodes

Figure 5.10. Estimation error ϵ_K

We can also integrate the reputation system and the extended MHT algorithm to design a complete robust tracking algorithm for sensor networks.

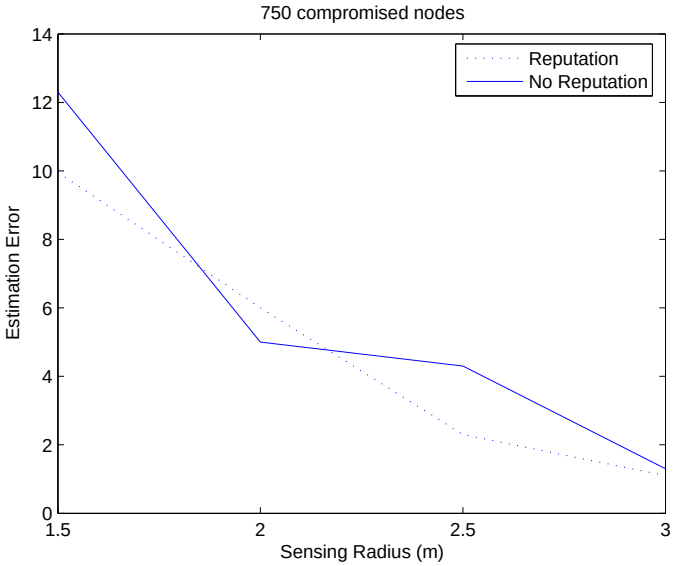


Figure 5.11. Estimation error ϵ_K.

Part 3

Application Security

Chapter 6

Sensor Networks in SCADA Systems

The Supervisory Control and Data Acquisition System (SCADA) refers to a large-scale, distributed measurement (and control) system. SCADA systems are used to monitor or to control chemical or transport processes, in municipal water supply systems, to control electric power generation, transmission and distribution, gas and oil pipelines, and other distributed processes. For example, the remote communications capability is quite advanced in SCADA systems, where fiber-optic cabling, satellite communications, bridges, and routers are routinely used for backup, redundancy, and capturing data from hundreds of remote terminal units and PLCs scattered all over the world to a desired central location. In one location, a manufacturer can capture 10,000 miles of pipeline system process data and distribute it daily to pipeline managers and corporate management[1][2].

To be very careful about the terminology, SCADA is not correct, but is the commonly used term for control systems. SCADA system monitors and controls a geographically dispersed process like and oil pipeline, which Distributed Control System (DCS) monitors and

[1] Permission to use parts of these works for the purposes of this dissertation have been granted by the IEEE and/or the appropriate copyright holder(s).

[2] Part of the contents of this chapter are based on [103].

controls a local process such as an oil refinery. Given many components and applications of SCADA and DCS are similar or identical, we use the term SCADA throughout this chapter.

A major drawback of typical SCADA systems is their inflexible, static, and often centralized architecture, accompanied by expensive cost of wiring devices together. Wireless sensor networking is a promising technology that can significantly improve the sensing capability of the SCADA system and significantly reduce the wiring costs.

SCADA systems are now leading the way for better control and monitoring of remote processes and plants through the use of new cost-effective hardware and software. Recently, a number of companies, prominently Dust Networks and Emerson Process Management, have teamed up to bring sensor networks in the field of process control systems. Their solutions offer a reliable, robust and secure sensor network that could interoperate with the existing wired and wireless devices of the SCADA systems.

The reason for the push to go wireless is twofold. First, using the wireless sensors enables the companies to gather more data from their deployment fields. In the past, a lot of remote locations were hard to gather information from since running wires for long distances is not always feasible. Wireless sensors give the industry an opportunity to gather more data regarding their processes, which in turn makes the system more efficient. Second, wireless sensors are an economically appealing solution since they eliminate the cost of running wires to all the field devices and decrease the installation costs.

However, SCADA wireless networks without security can be dangerous. Without user or device authentication, an adversary can send false data sensor data to the controller of a system from a neighboring area. The controller will then take an incorrect action, leading to diverse consequences, such as overflow of water tanks, or chip manufacturing errors.

In this chapter, we focus on the problem of intrusion detection and key management in SCADA systems. The rest of the chapter is organized as follows. Section 6.1 gives some background information on the typical architecture of SCADA systems and the integration

of sensor networks into these systems. Section 6.2 gives the motivation for developing the IDS and key management protocol. Section 6.3 describes the WirelessHART standard which will be used as the baseline for our proposals. Section 6.4 describes our model-based IDS, secure rekeying, and secure software update protocol. Finally, Section 6.5 discusses the future research direction.

6.1 SCADA Architecture

The general composition of the SCADA system includes: input and output signal hardware, controllers (intelligent electronic devices), Human Machine Interface (HMI), networks, communication, database and software. The bulk of the data acquisition comes from the Remote Terminal Units (RTUs). The RTU consists of Intelligent Field Devices (IFD), for example the controller devices or wireless devices. In the sensor network deployment, the wireless devices are sensor motes. Figures 3.1 (a,b) show the integration of wireless sensor networks with SCADA systems. The data collected by the IFD is then compiled and formatted in such a way that a control room operator using the HMI can make appropriate supervisory control decisions. The other devices in the field are gateways which are the access point of the the RTU with the process automation network.

The communication between the control devices and data acquisition devices generally follows a master-slave protocol, meaning, the slave devices only respond when being polled by the master device. However, in the case of an unusual event, the *notification by exception* is also allowed.

The SCADA system have a number of characteristics which distinguish them form the typical IT system. For example, availability and real-time response take precedence over confidentiality of the data, and the system has a long life time requirement. These characteristics result in different prioritization of the security requirements in SCADA networks, meaning

the security properties, in the order of decreasing importance, are: 1)Availability, 2) Integrity, 3) Confidentiality, 4) Privacy.

Given these requirements, the main goal of the attacker is then to compromise the availability and integrity of the gathered data.

6.2 Motivation

The security of SCADA systems is crucial given the criticality of their application. A successful cyber attack on the SCADA and other control system could result in massive cost and potential loss of life. Security studies from the U.S. Department of Energy (DOE) and commercial security consultants, including KEMA, have demonstrated the cyber vulnerabilities of control systems. More than 60 identified (though not publicly documented) real-world cases have occurred where electronic means have impacted the control systems' reliable operations. Control systems have been designed with specific reliability and availability requirements but not specific cyber security requirements. Adapting cyber security to these requirements is another challenge, and one we must meet.

If malicious actors could access these systems, they would have access to operational data critical to the operation of the system. Also, a knowledgeable attacker could modify the data used for operational decisions, the programs that control critical industry equipment, or the data reported to control centers. The impact could be destructive.

Such attacks could exceed equipment design and safety limits and potentially result in damage, premature system shutdown, and interference with safety system operations. Or they could immobilize control equipment. Consequences could include endangerment of public health and safety, environmental damage, or significant financial impacts due to loss of power production, transmission, or distribution. The following is a list of the general attacks possible on these systems [**?**], [104], [105]: 1)Gain access to the SCADA system, 2)

Identify the devices on the network, 3)Disrupt the master-slave communication, 4) Disable the slave, 5) Read and write data to the slave, 6) Program the slave, 7) Compromise the slave, 8) Disable the master, 9) Write data to the master,and 10) Compromise the master.

Control systems are susceptible to attack because they weren't designed to meet cyber threats. As control systems move from traditional closed networks into highly interconnected heterogeneous networks, containing both standardized and legacy technology, the threat environment has changed. In light of these heterogeneous networks, we cannot directly apply security technologies designed for common business IT systems to control systems and still provide adequate protection. Designers of control systems did not design them with operational requirements for reliability and availability. Hence, the integration of security into those systems must account for these requirements, which can differ drastically from those of business IT systems.

Sensor networks in SCADA systems are an emerging area, and there has been very little research done in the area of security of sensor networks used in SCADA systems. In the remaining parts of this chapter, we attempt to design 1) a model-based intrusion detection system, and 2) a key management and secure software update for sensor networks deployed in SCADA system. We use the WirelessHART [54] as the basis for our IDS. The details of this standard are given in the next section.

6.3 Background

In this section, we discuss the details of the WirelessHART standard, as well as describing some essential information on intrusion detection systems.

6.3.1 WirelessHART

WirelessHART is the first open and interoperable wireless communication standard designed to address the critical needs of the process industry for reliable, robust and secure wireless communication in real world industrial plant applications. The WirelessHART protocol operates according to the master-slave method. Any communication activity is initiated by the master, which is either a control station or an operating device. The slaves never send without being requested to do so, except for ”notification by exception” in case of an emergency event. The slaves respond only when they have received a command message from the master unit.

WirelessHART uses a wireless mesh network, meaning all the field devices can perform routing. In addition, the redundant paths make the network reliable since the messages can route around obstacles and hot spots. Other desirable features of WirelessHART include [54]:

- Compatibility with the existing wired devices
- Frequency hopping for added reliability and minimizing interference
- Multiple transmit power levels, which can be configured by the network user
- Clear channel assessment
- Multiple message modes, such as event notification, block data transfer, and acyclic request-response
- Security through using the AES-128 ciphers and keys

Figure 6.1 shows how various nodes running WirelessHART communicate with each other and the gatewaynetwork manager. The network manager resides on the gateways and is in charge of coordinating the communication schedules, key management, and monitoring the physical health reports.

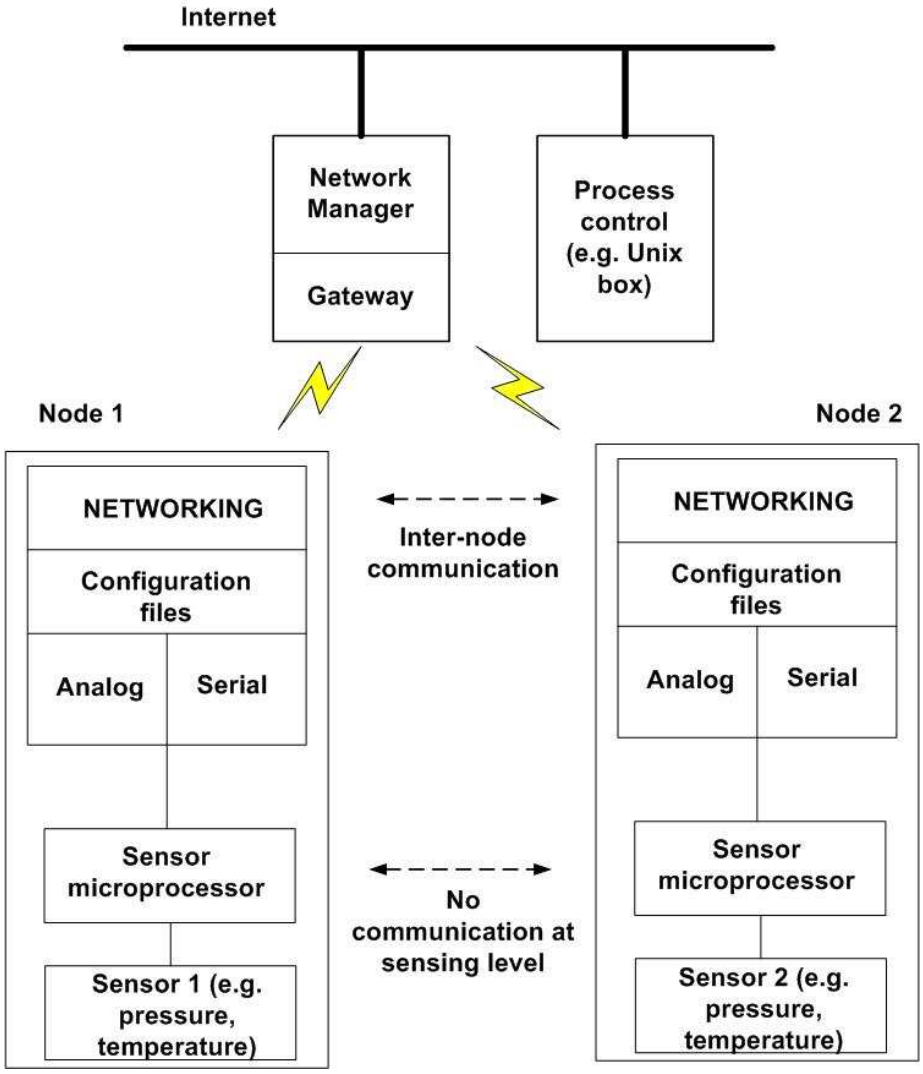


Figure 6.1. Node-to-node and node-to-gateway communication.

6.4 Proposed Solutions

In the following subsections, we discuss two solutions for increasing the security of sensor networks uses in the SCADA systems: 1) A model-based intrusion detection system, and 2) Key management and secure software update protocol.

6.4.1 Multi-Layer Model-Based IDS

Generally IDS consists of the following components [106]:

- Sensors: to detect events
- Analysis center: analyzes the data gathered by the sensors
- Databases: helps the analysis and response systems
- Response center: formulates an appropriate response to an event

In the literature, intrusions have been categorized into two types[3]:

- *Signature-based detection*: in this scenario, the system looks for the known attack patterns and tries to match the so called *signature* of the attack to identify these intrusions.
- *Anomaly detection*: in this type of intrusion detection, the normal user behavior is defined and the system looks for any significant deviation from these established normal uses.

An intrusion detection model consists of two components: the *feature space*, also known as attributes, and the *modeling algorithm* which uses the features to identify the intrusions. The most important task in building an intrusion detection system is defining a feature space that can accurately represent the normal and anomalous behaviors of a user.

[3]For a more comprehensive taxonomy of the intrusion detection systems, we refer the reader to [106].

Our solution focuses on the second type of an IDS, meaning the anomaly detection which is based on defining model of normal system behavior. Our IDS defines normal behavior at different layers of the network stack, as will be explained in detail in the following subsections.

6.4.2 Assumptions

The communication between the sensor nodes and the network manager is over a wireless medium, and we therefore adopt the Dolev-Yao attacker model, where an attacker can eavesdrop, intercept, modify, or inject messages into the wireless communication [61].

Moreover, we assume that an attacker is not bound to the computation power of a node but can use more powerful devices. The attacker can also get physical access to a node and learn all the secrets stored in the compromised node.

We also assume that the network manager is a well-protected unit, and is not susceptible to intrusion or denial-of-service attacks. In addition, the network manager is a powerful device and does not have the same restrictions as the nodes in terms of CPU power, memory size, or transmitting power.

6.4.3 Problem Definition

Our IDS consists of two components: a central IDS, and multiple IDSs distributed in the field among the sensor nodes, which we call *field IDS*. The main IDS resides on the network manager and is responsible for monitoring all the packets that arrive from the sensors and are destined for the masters. In addition, this central IDS monitors the traffic from other connected networks, such as the Internet; however, the focus of this chapter is on the sensor network portion of the system. We emphasize that our work does not focus on designing rules for intrusions from the Internet since the traffic from the Internet requires a separate set

of rules for intrusion detection. The rules we develop here are specifically for sensor network traffic and are not suitable for Internet traffic. The field IDSs are deployed using 'supernodes', which are sensor nodes with higher communication and computation power and have tamper-resistant hardware. The reason for choosing supernodes instead of regular sensor nodes for a field IDS is that supernodes are harder to attack and to extract the cryptographic keys from.

The field IDSs are responsible for *passively* monitoring the communication of the sensor nodes in their neighborhoods to collect trace data. They periodically send monitoring messages to the central IDS, where the messages contain information on the traffic patterns of sensor nodes in their vicinity.

We chose this hybrid IDS architecture mainly because of the characteristics of the SCADA system. Fully-distributed IDSs have been proposed in wireless sensor networks, such as [107]. However, if each node were to run an IDS, it would have had to spend more energy monitoring its neighbors and drawing inferences based on its observation. It is also much more complicated trying to arrive at a global decision when using a fully distributed IDS system. The nodes have to pass information among themselves to arrive at a consensus. The fully distributed scheme is also more vulnerable to attacks since the software is running on the nodes that are not tamper resistant. As mentioned in Section **??**, it is fairly easy to compromise a subset of the nodes physically and subvert their operation. This will nullify the whole point of using an IDS to monitor the behavior of the sensor nodes.

Operation of the central and field IDS consists of three phases:

- *Data gathering*: The field IDS listens in the promiscuous mode to the sensor traffic in its neighborhood and gathers information. The sensors send *physical health monitoring* reports periodically, for example every 15 minutes. Then, the field IDS extracts relevant information, such as the packet fields from these reports. In addition to the information gathered by the field IDS, the central IDS gathers trace data from the incoming packets (from the gateways) and the field IDSs. This provides the overall IDS

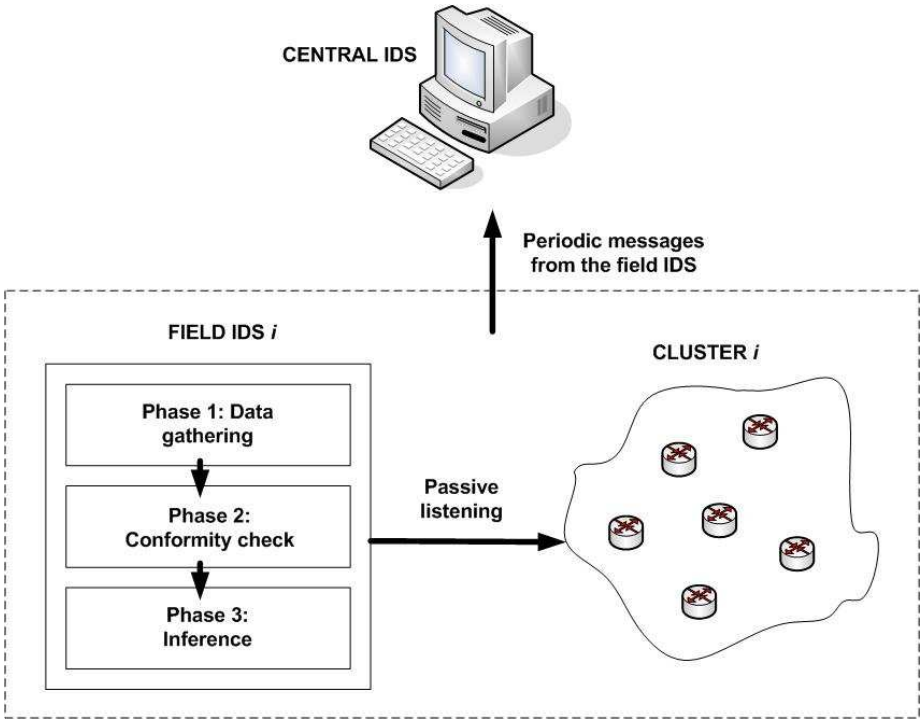


Figure 6.2. The IDS layout.

with more information given that the central IDS has a global view of the network. This information is used in the next phase.

For analysis purposes we have several fields in the physical health reports. The reports contain information about at most N neighbors that have been heard in the last 15 minutes. This includes neighbors that are part of normal scheduled communication, and neighbors that are overheard during the infrequent 'hello world' broadcasts that every node makes. The information about each neighbor includes the number of TX attempts, number of ACKs received, number of RXed packets, and number of bad CRC and bad MIC packets received.

- *Conformity check*: Both IDS modules use the extracted data from phase 1 to perform analysis and check for conformity of the node behavior to the normal behavior.

- *Inference*: After the analysis is done, the result is sent to the inference unit for deciding whether a detected anomaly is a transient network failure or a malicious attack. For

this phase to make accurate inferences, the IDS must keep a 'state' or history for each node it is monitoring to ensure that the system is capable of distinguishing occasional network failures from real attacks, at some confidence level.

Figure 6.2 shows the schematics of our proposed IDS.

The IDS also needs to monitor for lying nodes. Given that the sensor nodes are unattended in the field, it is conceivable that some of the nodes could be physically compromised. An attacker who has access to a node can send wrong/bad observations to the data-gathering station. Therefore, it is important that the IDS checks for invalid or anomalous observations. This can be achieved by comparing the data gathered from clusters of sensor nodes in the same physical neighborhood. The network manager has a global view of the network topology, and therefore is capable of partitioning the sensor observations into neighborhood clusters.

Our IDS monitors the network at various levels: the physical layer, link layer, traffic pattern, and communication datagram. In the following subsections, we will explain each of these in more detail, but first we state our assumptions regarding the communication among different IDS components.

We assume that the communication between the central IDS and the field IDS is encrypted using an end-to-end network key to ensure confidentiality and integrity of the messages. In addition, the packets transmitted among sensor nodes have their payload encrypted and have a message integrity code (MIC) on their headers. However, the packet header is not encrypted so that the field IDS can listen to the sensor node communication in promiscuous mode. Moreover, we assume that there is a mechanism in place for authenticating the field IDSs to the network administrator if there is the need to do so, such as using a digital signature.

Physical Layer:

The sensor nodes generally transmit at a pre-configured power level. In some scenarios,

the nodes can adjust their transmission power to different prescribed levels, but the levels are still known to every node and the administrator since it is a configuration parameter. Therefore, the IDS can have a rule that would monitor for normal levels of transmission power, and if an attacker attempts to jam the network, the IDS would raise a flag. Jamming is the interference with the Radio Frequency (RF) used by the nodes in a network. It makes use of the broadcast nature of the communication medium.

Data-Link Layer:

The data-link layer and the routing layer specifications of WirelessHART standard have been based on the Time Synchronized Mesh Protocol (TSMP) from the Dust Networks [108]. In TSMP, the devices use designated timeslots and frequency hopping for communication. Therefore, the IDS has to monitor the nodes for adhering to the correct frequency sequence and timeslots at all times. Otherwise, a compromised node can cause numerous collisions at the MAC layer by transmitting at a wrong frequency or in a wrong time-slot. Therefore, there should be a mechanism by which the field IDS keeps track of transmission schedule of different nodes.

It is also possible that the attacker would change the sender field of a packet to match the time-slot it is sending the packet in. This strategy will make the attack undetectable to the monitoring system. However, it will require the attacker to have access to all the keying material so that he can encrypt the packets with the correct link and end to end keys.

Routing:

Given availability is the most critical requirement of a SCADA system, the routing protocol needs to ensure that the packets sent by sensor nodes arrive at the master station consistently and at all times. In order to guarantee this availability, WirelessHART standard suggests using fully mesh routing. Fully redundant routing requires both spatial diversity and temporal diversity. TSMP covers spatial diversity by enabling each node to discover multiple possible parent nodes and then establish links with two or more, and the temporal diversity

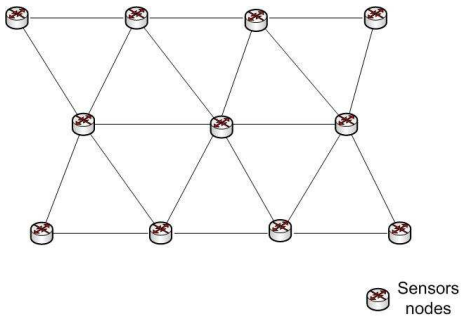


Figure 6.3. Fully mesh routing schematic.

is handled by retry and failover mechanisms. Figure 6.3 shows an example of mesh routing topology.

Each node in the TSMP implementation maintains its own neighbor list. This neighbor list consists of parent nodes and child nodes. A node may have as many parents as required, i.e. this is a configurable parameter. As mentioned before, the node message transmissions occurs in one time-slot and on one frequency (TDMA and frequency hoping). In each time-slot a message is sent, and the sending node switches to receiving mode and waits for an acknowledgement (ACK). Should an ACK not arrive within the time-slot, the sending node will retry in the next available slot. This will generally be to an alternate parent and will always be on a different frequency [108].

TSMP makes use of NACK packets as well which are messages indicating the expected packet was not properly received by the neighbor node. If a NACK is received by a node, then the sending node will retry on the next available slot. NACKs are generated due to a number of reasons: invalid checksum (FCS), invalid message integrity code (MIC), or a full message queue by the receiving node.

Each TSMP node keeps track of the missing ACKs and NACKs. In the event that there are a number of transmissions without acknowledgement, the sender node will consider the path invalid and initiate communication with the next available node on its neighbor list.

Traffic Characteristic:

To issue a live incident response, the IDS must collect and analyze volatile and nonvolatile network traffic data, in addition to the previously mentioned factors. For our purposes, traffic characteristics consist of two components: traffic load and traffic pattern. Traffic load is concerned with the amount of information being passed around in the network. In the WirelessHART specifications, for example, the maximum payload of the packet is 127 bytes. In addition to the packet size, the load is concerned with periodicity of the transmission of packets. The sensors send their monitored data periodically to the master (i.e., network manager), for example, every 30 seconds. Moreover, the 'physical health reports', which carry the statistical information of the wireless channel, are sent once every 15 minutes from each node. Therefore, the traffic load can be approximated fairly accurately. If the IDS observes a much higher/lower traffic load than expected, or if the packets are larger than 127 bytes, it could raise a flag for suspicious activity.

Traffic pattern refers to the communication pattern among the nodes in the network. For example, based on the discussion of the SCADA system layout, we can identify the following patterns: 1) Master to slave, 2) Network manager to field devices, 3) HMI to masters, and 4) Node-to-node communication at the network level, to route packets to the master.

The IDS can monitor the traffic for legitimate patterns. If at any point it observes unusual traffic, such as node to node at the sensing level (and not the network level), it can raise a flag or can monitor the activity of the node more closely.

Communication Datagram:

The communication datagram specifications in WirelessHART give the details of the required fields in a communication packet. It specifies the format of each field, and the valid range of values for each field.WirelessHART datagram specifications include [54]:

- Command priority: the IDS needs to monitor that the sensors do not send low-priority messages as high priority.

Physical layer	Link layer	Network layer	Security	Transport layer	Application layer
Preamble, Delimiter, Byte count	Source, Destination, Network ID, etc.	Source, Destination, etc.	Security controls, MIC, etc.	Transport control, etc.	Byte count, Data, etc.

Figure 6.4. Packet format defined by WirelessHART **??**.

- Data type: the data is command, process data, alarm or normal.
- Packet type: acknowledgement packets that include timing information to synchronize the TDMA (time-division multiple access) operations.
- Bandwidth request: this is used to provide elasticity of event traffic and ad hoc request/response maintenance messages.

Figure 6.4 shows a generic communication packet format supported by WirelessHART specifications.

Policy Rules for Attack Detection

We outline our IDS policies, discuss which attacks they are targeted for, and which IDS (central or field) implements these policies. We assume that there is an access control list at the network manager, and the nodes have unique link keys, so that there is no need to define policies for the Sybil and cloning (replication) attacks.

Policy rule 1: Monitor packet count for each node in terms of the byte size of each packet and frequency of transmission. As pointed out in the previous section, the communication datagram is well defined with respect to the packet size and number of bytes. In addition, the slaves, i.e., sensor nodes, report to the master station in preconfigured intervals, for example every 30 seconds. This rule is applied at the field IDS and the central IDS and prevents the compromised nodes from sending too much or too little traffic. Especially if a compromised node tries to cause collision by sending a lot of traffic, the IDS is able to monitor the node's activity level, and raise a flag if necessary.

Policy rule 2: Monitor physical health reports in order to determine if they are transmitted in the preconfigured intervals. For example, the current implementation has the nodes send a physical health report once every 15 minutes. This rule is implemented at the field as well as the central IDS, and helps prevent nodes from injecting too much (or too little) traffic into the network.

Policy rule 3: Monitor the request/reply traffic between the slave (sensor nodes) and the master station. The normal traffic pattern, as mentioned earlier, is a request from the master followed by the reply from the nodes, and in few cases a 'notification by exception'. Therefore, the IDS keeps track of how many requests/replies it sees in a given interval, which is a known number, and how many 'notification by exception' packets. The request, reply and notification by exception packets are set with a flag in the packet header. If the requests are coming from any device other than the master (or in some cases a handheld device), then the IDS should raise a flag. Also, if a node sends too many exceptions, the IDS should notify the system engineer. This policy can be applied at both the field and central IDSs.

Policy rule 4: Monitor the individual datagram fields. The WirelessHART specifications give guidelines on the valid entries of each packet field. Therefore, the IDS must monitor the contents of each received packet to ensure valid entries for each item. For example, even though there is a field for sending control messages[4] to the sensor nodes, it is not used by any currently deployed implementation. Therefore, the IDS should flag any packet that has the control option set. This rule must be implemented at the central IDS because the communication datagrams are encrypted, and the field IDS is not able to decrypt the contents of the packet in order to examine the packet fields. This rule protects against attacks in which the compromised node sends packets with invalid fields or contents.

Policy rule 5: Monitor the layer at which the communication among nodes is performed. As explained earlier, the communication among the sensor nodes is at the network layer in

[4] *Control* in this context means using the sensors as actuators to adjust a valve for example.

order to perform routing, and not at any other layer. This means that the sensors do not directly exchange information with each other, but rather use their neighbors for the purpose of routing their packets to the master. If a node attempts to communicate at any other layer with its neighbors, then the IDS should raise a flag to notify the system. This rule is implemented at the field IDS since the central IDS cannot monitor the local communication of the sensor nodes because of the communication range limits. However, the central IDS may correlate messages of this type from the field IDS components and infer the extent of unallowed communication.

Policy rule 6: Monitor the power level of the node communications. Since there are preconfigured levels of transmission power for the deployed nodes, the IDS can monitor for any deviation from the accepted levels. This rule is implemented at the field IDS and is designed to guard against a jamming attack. Again, the central IDS can use correlation to infer the extent of the jamming. Even though jamming is hard to defend against, it is useful for the network personnel to know when their system is under a jamming attack.

Policy rule 7: Monitor frequency/channel hopping sequence. Based on the WirelessHART protocol, the sensor nodes agree on the frequency hopping sequence for the time slots in which they communicate with each other. Therefore, the IDS needs to monitor the nodes to ensure that they follow the agreed-upon sequence and do not deviate, which will result in high number of packet collisions. This rule must be implemented at the field IDS level.

Policy rule 8: Monitor the MAC layer authentication at the field IDS. If a packet fails the authentication, it may be due to an attacker injecting bad packets into the network. Therefore, the IDS can signal a possible intrusion.

Table 6.1 summarizes rules used for each layer and threat.

For the lying nodes, the solution is to keep a history for all the reporting nodes at the network manager. If there is a spike in the reported values, look at the neighbors' values, and flag if it is an anomaly. An example of this work is [8] where the authors use a reputation

Table 6.1. Policy rules for each layer and threat. NA refers to Not Applicable.

	Threat 1	***Threat 2***	***Threat 3***	***Threat 4***	***Threat 5***	***Threat 6***
Physical layer	Rule 6	NA	NA	NA	NA	NA
Data link layer	NA	Rule 5	Rule 1,2	Rule 7,8	Rule 8	NA
Network layer	Rule 1	Rule 3,8	Rule 1,2,3	Rule 2,4, 5	Rule 4,5,8	Rule 3

system to track the nodes behavior over time. The anomalies and node misbehavior are determined locally by comparing the values reported by neighboring nodes.

Adding the IDS into the network requires very little overhead. As discussed earlier, the health reports are configured to be sent by the sensor nodes in the current implementations. Therefore, the only overhead is adding the extra supernodes that could be used for the field IDSs.

Intrusion Response Mechanism:

After detecting an intrusion or misbehavior, it is essential that the system takes appropriate action to stop the attack. Passive responses are typical in the IDS, such as logging of the information, and real-time notification. However, for the IDS to be effective, especially in the context of SCADA applications, we must have an *active* response system. Given the lack of physical security in wireless sensor networks, it is essential that the IDS has an effective active countermeasure.

The type of intrusion response depends on the level of confidence in the intrusion, and the risk that the attack (or intrusion) carries. Defining the threat level of each attack is dependent on the type of the SCADA network application (e.g., oil and gas, electric power), and can be identified by a 'threat score'. Determining the threat score is beyond this work. We assume that the risk analysis is done and the threat scores are loaded into the IDS by the network administrator.

Based on the threat score and the confidence in the intrusion detection[5], some of the possible responses are: 1)Remove the compromised nodes from the network, 2)Rekey the network to secure the communication channels, and 3) Reauthenticate the nodes.

When the IDS suspects that an outsider has picked up the frequency hopping schedule of the nodes and is jamming a portion of the network, the central IDS can send a request to the nodes, possibly via the network manager, to change their frequency hopping sequence. This will prevent the attacker from continuing to jam the network by using the old frequency hopping sequence.

6.4.4 Key Management and Secure Software Update

Sensor motes used in the current SCADA environments are preloaded with unique link keys during the manufacturing phase, and an access list is provided to the network manager prior to deployment which contains the link keys along with the node ID [54]. Given the initial key establishment is performed prior to deployment, it is not considered a problem in these networks. However, to avoid using the same key for a long period of time, generating new keys is a necessity. In fact rekeying is suggested as part of the new standard proposed for wireless SCADA environments [54] which we described in Section 6.3.1. In this section, we describe our protocol for rekeying and secure software update. The rekeying protocol provides both backward and forward secrecy. Forward secrecy guarantees that a passive adversary who knows a contiguous subset of old group keys cannot discover future group keys. On the other hand, backward Secrecy guarantees that a passive adversary who knows a contiguous subset of group keys cannot discover the old group keys.

There has been extensive research on secure communication and key management in wireless sensor networks, a complete survey of which could be found in [31]. Beaver et al. [109] present a model of a SCADA network and discuss relevant key management proper-

[5]For example, with x% confidence, nodes X, Y, and Z are compromised.

ties for such a network. There are, however, a number of prerequisites and assumptions that must hold in order for this key management scheme to be effective. For example, a long-term key is stored in these devices and used to generate new keys. However, it is not clear how the long-term key is protected against node compromise in these papers.

TinySec [15] offers efficient link-layer authentication and encryption for wireless sensor networks. It can be used with a network-wide key that is loaded in the nodes prior to the deployment. TinySec can also be used with group keys or pairwise keys. However, the process of key distribution is not described in this work. In addition, TinySec offers no replay protection.

MiniSec [110] is a secure network layer that provides data secrecy, authentication, and replay protection. However, the security in MiniSec comes from using encryption keys stored on the nodes. The process of key establishment in this protocol is not revealed.

TinyPK [111], in contrast to the previous approaches, uses public key cryptography to secure sensor networks. This approach allows authentication and key agreement between a sensor network and a third party as well as between two sensor networks. However, this work does not discuss any rekeying between sensor nodes and the network manager.

SPINS [4] is a suite of security protocols that contains two components: SNEP and μTesla. SNEP provides data confidentiality, authentication, and freshness. μTesla provides authenticated broadcast for wireless sensor networks. The security of the protocol relies on a master key that is shared with the base station. Every other cryptographic key that is needed is derived from this master key.

Deluge [112] provides over-the-air programming for TinyOS without any guarantees on the security of the reprogramming. Dutta et al. [113] present a secure extension to Deluge using digital signatures and hashes to verify authentication, integrity, and freshness. On the down side, this approach does not provide confidentiality and relies on lower-level encryption

mechanism, such as TinySec. In addition, the software binary is not kept secret, since any device in the network can access it.

6.4.5 Assumptions

We make a number of assumptions when designing our key management and secure software update schemes which are listed below:

- The network manager has a public and private key pair, and the public key of the network manager is preloaded onto the sensor motes.
- Shared unique symmetric keys (link keys) between each node and the network manager are pre-installed in the nodes and the network manager. The link key is actually split into two keys, where one is used for authentication and the other for encryption.
- A global key (network key) is preloaded in the nodes and the network manager.
- Different types of sensor nodes are deployed in the network. Therefore, they require different types of software to operate. In order to update these heterogeneous softwares, we use the multicast approach. This requires having a way to establish a group for the subset of nodes that need software updating.
- When a new node joins the network, it uses a unique link key to authenticate itself to the network manager. The link key is also used for encryption and providing integrity of the communication.
- A network-wide key is used to prevent outsiders from injecting packets into the network. The network key also provides link level (per-hop) message authentication. The message authentication code (MAC) is verified and recalculated on each node prior to forwarding it to the next node.

- Each node contains a counter, which is used for generating random numbers. The random number is generated through a pseudo-random generator constructed as follows. Each node is preloaded with a random key generated with strong randomness prior to deployment. This key is kept secret and used together with the function F, which takes as input the counter. $random\_number = F_{key}(counter)$ The counter is incremented after every use. The F function can be implemented as a hash-keyed MAC (HMAC). The generated random number is used in the key management, as a nuance, to prevent replay attacks.

- Communication between a node and the network manager is used for establishing and distributing keys, as well as reading sensor data and sending control requests. The network manager communicates with groups of nodes for remote software updates.

- We assume that mesh routing is used in the wireless sensor network in the same manner suggested in WirelessHART [54] and Time Synchronized Mesh Protocol (TSMP) [108]). Mesh routing provides robustness and reliability for the data communication.

6.4.6 Problem Definition

We formulate a definition of the problem and the security requirements for a complete solution. A secure and efficient way to establish and regenerate cryptographic keys in wireless SCADA is needed. Moreover, a secure way to update the software on nodes in the wireless SCADA is needed.

Security Requirements

The security requirements that we are concerned with when designing the key management and secure software update protocol are: 1) Data confidentiality, 2) Data integrity, 3)

Data authentication, and 4) Data freshness. Data authentication can be achieved using a MAC, calculated using a shared symmetric key, on the communicated data. This works well for communication between two devices. However, when several devices are involved, any device knowing the MAC key can impersonate the sender and forge messages to the receivers. Therefore, for multicasting of software updates, an asymmetric authentication approach is necessary. Data freshness provides protection against replay attacks. This means that during the key establishment phase the protocol must ensure that the messages are fresh. Finally, we have to define a *secure channel*: a channel that provides data confidentiality, integrity, authentication, and freshness.

6.4.7 Design Goals

Here we outline our design goals for the key management and software update protocol. These goals are based on the security requirements that we highlighted in the previous section:

1. If an attacker compromises a node, the attack should not spread to other nodes. Therefore, a global key can not be used for authentication or encryption. Instead, there has to be a pair-wise link keys among nodes and the network manager. An attacker who compromises a node should not be able to masquerade as another node nor decrypt communication that is meant for other nodes.

2. The key establishment should be immune to man-in-the-middle attacks. This includes both the initial key establishment and the subsequent rekeying. This property ensures that an attacker cannot sit in the middle and establish keys with the nodes and the network manager, pretending to be the other.

3. Data between a node and the network manager should be sent over a secure channel, as defined earlier. Each node needs to establish link keys with the network manager.

4. Rekeying should provide backward secrecy and forward secrecy.

5. Software updates should be sent over a secure channel to a group of nodes. The nodes should verify that the software is fresh and comes from a trusted source. In addition, an attacker should be prevented from gaining access to the proprietary software binary.

6. Network manager authentication by the nodes is necessary. This is achieved by using individual link keys between the nodes and the network manager. For software binaries, this is achieved by generating signatures on the network manager and having the nodes verify the signature.

7. There is a need for both device-level and link-level authentication as well as data authentication. A network-wide key is used for link-level authentication. This is used for calculating a MAC on a per-hop basis. Device authentication is achieved using the individual link keys that are stored in the motes. Additionally, these keys are used for end-to-end data authentication.

6.4.8 Key Management

In this section, we describe the details of our key management scheme for the link and group key establishment. The link keys are used to provide confidentiality and authentication on the data traffic. The group keys are used in multicasting for providing confidentiality of the software binaries. Finally, public key cryptography is used to verify the integrity and authenticity of the binaries.

In order to better understand the protocols in this section, the notation used in the protocols is outlined in Table 6.2.

Table 6.2. Notation used in the key management and secure software update protocols

A, B	Nodes A and B
NM	Network Manager
N_A	Nonce generated by A
PK_{NM}	Public key of the network manager
SK_{NM}	Secret key of the network manager
K_{NMA1}	Symmetric key shared between A and NM
K_{NMA2}	Symmetric key (rekeyed) shared between A and NM
$h(x, y)$	One-way hash function h with input x and y
$E_K(M)$	Encryption of message M using the key K
$MAC_K(M)$	Message authentication calculation of message M using the key K

Link Key Establishment

Here we describe the rekeying of a symmetric key between the network manager and a sensor mote. This procedure is shown pictorially in Figure 6.5. A node initiates the rekeying with the network manager by sending a rekeying request with a random number encrypted using the public key of the network manager. This is called *message 1* in the figure. The rekeying by the node can be done on daily or hourly basis depending on the perceived threat to the system. In order to provide forward secrecy against a passive attacker, we encrypting the random number with the public key. A passive attacker will not be able to decrypt the random number and calculate the following link keys even if he can get access to the current link key through brute force methods.

It should be noted that all messages contain a MAC, which is calculated using the payload and the current link key. This provision prevents an attacker from injecting a random number encrypted with the public key of the network manager to initiate the rekeying procedure in an impersonation attack. This attack will only succeed if the attacker possesses the current

link key for that particular node. In addition, MAC prevents an attacker from modifying the contents of a message[6].

The new link key is derived through a one-way hash from the current link key and the random number to provide backward secrecy. An active attacker that records messages and compromises a key K at some point in time cannot decrypt messages that were encrypted with earlier keys (K-1, K-2, .. K-N). Moreover, an attacker cannot reconstruct earlier keys since a one-way hash function is used to generate the new keys. This rekeying protocol is based on Cebolla [114]. The rekeying process is as follows: during the rekeying phase, each side has a set of old and new link keys. The node first decrypts with the old key and switches if decryption fails. The network manager tries to decrypt with the new key first and switches if decryption fails.

If the network manager did not receive the rekeying request, it continues to use the old key to encrypt messages. Also, if there are any old messages that have been delayed due to communication delays, the messages are encrypted with the old key, which is shown by *message 2a* in the figure. If the network manager received the rekeying request, it sends the new message encrypted with the new key, i.e. *message 2b*. The node tries to decrypt using the old key first. If the decryption is successful, then it continues using the old key until decryption fails. If decryption fails, the node tries to decrypt with the new key and if the MAC is verified successfully, the node deletes the old key and use the new key. If the node received a message encrypted with the old key, the node continues to use the old key when replying to the network manager, as shown by *message 3a*. If the node is using the new key, the next message is encrypted with the new key. The network manager first tries to decrypt the received message with the new key. If it is successful, it deletes the old key. If it fails, the network manager uses the old key to decrypt the message and tries to decrypt using the new key on the following messages again until it succeeds.

[6]Each message looks like $payload, MAC_K(payload)$.

Instead of deleting the old key instantly after receiving a message encrypted with the new key, a time window can be used during which both keys are active. The size of the time window depends on the protocol being used and on the wireless medium characteristics, such as average delay on a link. This allows delayed old messages to be received properly to a certain point.

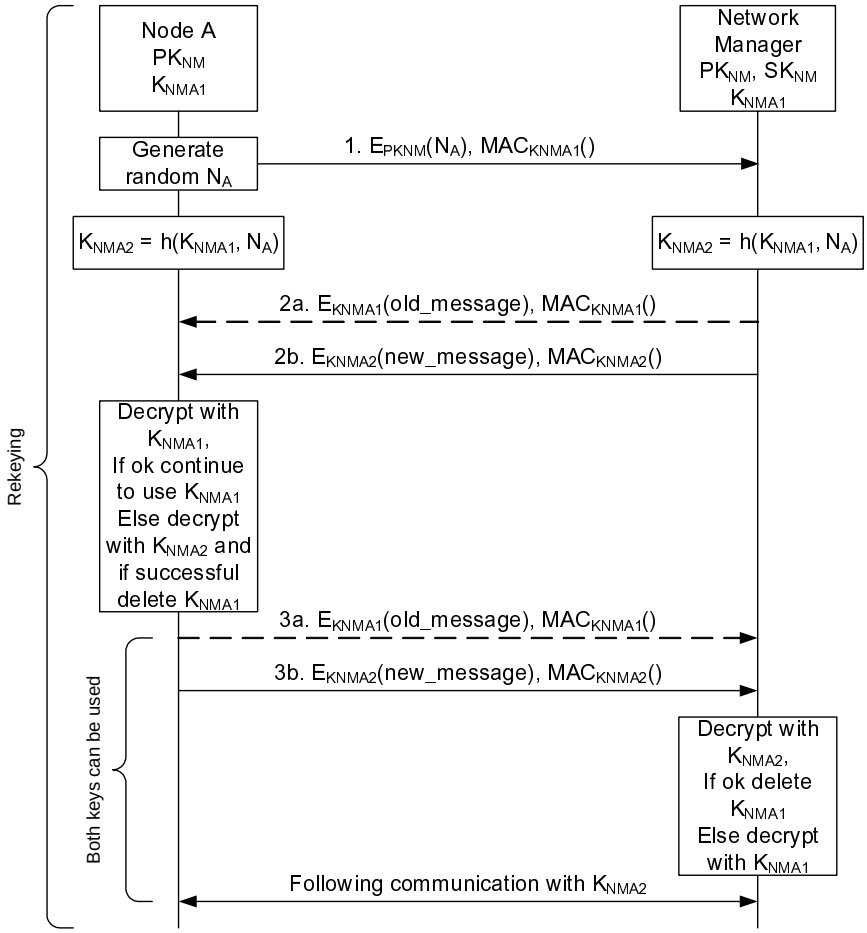


Figure 6.5. Between network manager and node.

Group Key Establishment

Here, we describe the details of the group key establishment process. The network manager uses the individually established link keys to distribute the group key. This protocol

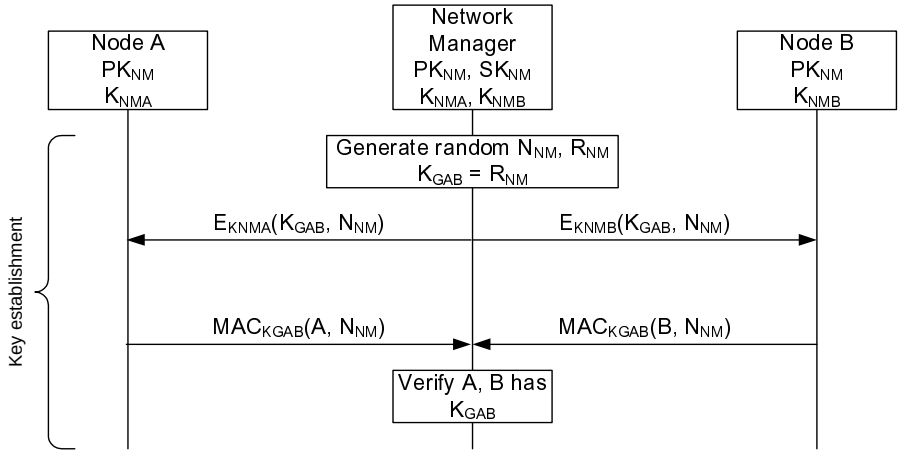


Figure 6.6. Group key for nodes A, B .. N.

is loosely based on a protocol for secure multicasting of software updates in future vehicles [115]. The details of establishing and distributing a shared group key is shown in Figure 6.6. First, the network manager generates the group key, which we call K_{GAB}. The group key along with a nonce N_{NM} is then distributed to each of the individual nodes. The nodes respond to the network manager when they receive the group key. To verify the group key, the nodes calculate an HMAC on their identity and the received nonce using the group key as the key. The network manager can then verify that the nodes possess the group key.

Secure Software Updates

Secure software updating is based on establishing a way for secure multicasting. Given different subsets of nodes need different types of software, each group must have its own group key. The network manager can then use the group key to send software updates for a particular group. Our protocols is based on an extension of the secure deluge network programming for sensor nodes [113] by encrypting the packets with the established group key.

The scenario is shown in Figure 6.7. First, the the network manager splits the new software binary into data fragments, which we denote D1 to DN. Then, the fragments are hashed

in reversed order. We denote the hashed fragments by HN to H1. Each hashed fragment is then stored together with the following fragment, resulting in a hash chain. Next, the hash of the first data fragment H1 together with X is signed by the network manager such that a node possessing the public key of the network manager can verify the authenticity and integrity of the first hash. X contains information about the software name, version, and the number of fragments (of the binary). The first packet contains the first hashed fragment; therefore, a node can verify all the future fragments by verifying the hash in each fragment (hash chain). This process enables a node to verify that the received software binary has not been modified by an attacker while in transition, and the binary does in fact come form the network manager.

The fragments H1 to HN contain a hash of the data fragment along with a random nonce, called N, which is selected by the network manager, i.e. $H1 = h(N, D1)$. This nonce helps prevent precomputing a pre-image collision [116] by randomizing the hash function. In addition, each data packet includes a header consisting of the software name, version, and the packet number in chain.

To provide confidentiality for the software updates, each packet is encrypted with the group key. This prevents any node that does not possess the group key from decrypting the packets.

The encryption mode used here is CBC (cipher-block chaining), which has been implemented in TinySec [15]. In CBC each block of plaintext is XORed with the previous ciphertext block before being encrypted. This causes each ciphertext block to be dependent on all plaintext blocks that come before it. For example, in Figure 6.7, the ciphertext C1 is calculated from the plaintext D1 and H2 together with the previous ciphertext C0. Other encryption modes have been suggested, such as OCB (Offset CodeBook) [117] used in MiniSec [110], counter-mode [118] used in ZigBee [119], and SNEP [4]. OCB provides authenticated encryption. However, in our scenario we do not need use this method since the hash chain

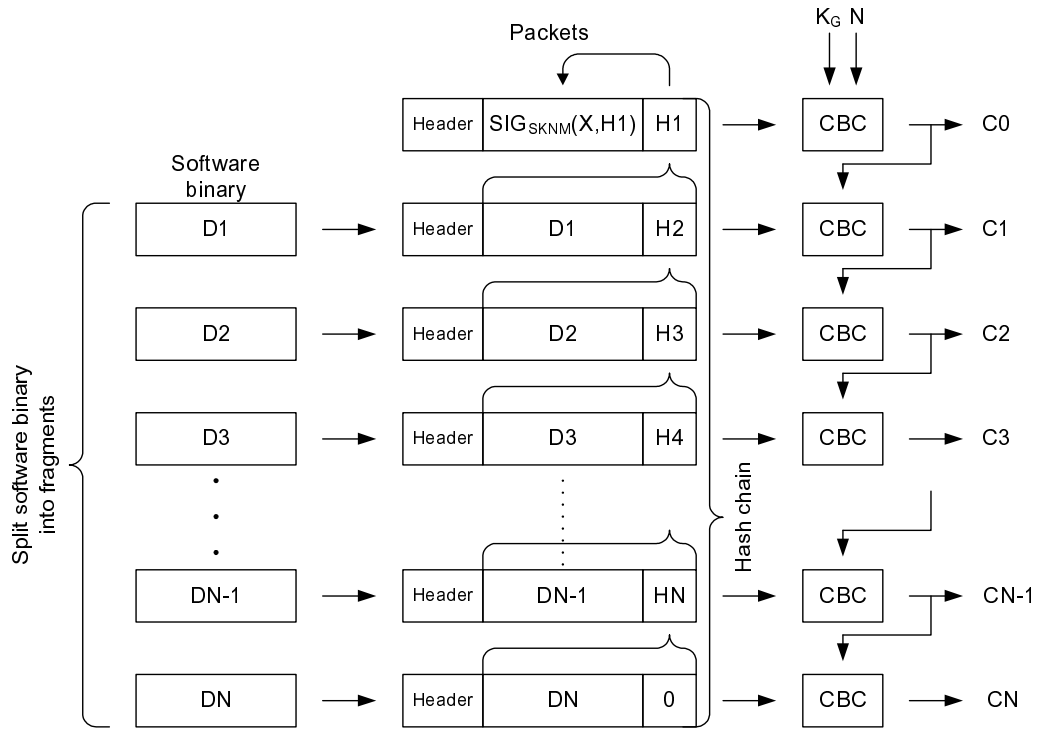


Figure 6.7. Secure software update procedure.

and signature provides authenticity. In addition, given counter-mode requires synchronized counters, it is not suitable for our multicasting scenario.

We argue that we do not need replay attacks for our software update scenario. This is due to the fact that each packet contains a hash of the following packet, and the motes can verify the order of the packets. If a packet is replayed at a later time, the packet number in the chain and the hash will not match, and the packet is discarded. If a packet is lost and an attacker replays that packet, this would help the distribution of the software instead of interrupting it. If an attacker tries to replay the entire software update process at a later time, the software version number in the replayed packets will be lower than the currently installed software in the nodes, and the nodes will simply ignore those packets.

6.4.9 Implementation Issues

In this section we discuss the feasibility of implementing the key management protocol we described in Section 6.4.8.

The maximum payload of the data packets is large enough. Therefore, the exchange of the nonce and keys in the rekeying protocol. Establishing a new key through traditional means, such as performing a Diffie Hillman key exchange, is computationally expensive. In addition, using such a key exchange methods is really not necessary since the nodes and the network manager share an initial secret.

The overhead for the secure software update is also practical for implementation given we use a hash chain and a digital signature on the first hash only. This reduces the number of public key operations to one per binary instead of one per packet. In addition, it provides authenticity and integrity of the entire software binary. This approach, to *receive-verify-store* on a packet basis instead of *receive-store-verify* of the entire binary, is efficient and well suited for wireless sensor networks since packets can easily be lost. Retransmitting the entire binary, as opposed to the missing portion, is very costly in sensor networks. This approach has been implemented and run on TinyOS in [113]. It is delay tolerant since there is no dependence on time synchronization. Moreover, incremental verification of received messages is achieved since each message contains a hash of the following message. To make this method suitable for multicasting in wireless SCADA, we extend their protocol by encrypting the packets with a temporary group key such that only the intended receivers can decrypt it. This prevents an attacker and nodes that are not intended to receive the software from passively eavesdropping on the communication and getting access to the new software binary.

Computational Overhead

It is important to ensure that the proposed key management protocol does not impose a great amount of computational overhead. To that end, we not that the rekeying protocol requires one public key encryption on the node to encrypt the random number and one hash operation to generate the new key. The group key establishment requires one symmetric key decryption and one MAC calculation on the node.

The software update process requires one public key operation to verify the signature for the first packet. It also took one symmetric key operation and one hash for each subsequent packet. According to TinyECC implementations on a Imote2 sensor node, a 128-bit ECC (Elliptic Curve Cryptography) signature verification takes approximately three seconds [120]. Given software updates are infrequent and require only one signature verification for the entire process, three seconds is a reasonable time delay for implementation. In addition, the cost of symmetric key and hash operations is negligible in comparison to public key operations which makes the the group key establishment very efficient. Finally, the rekeying and software update process require one public key operation each which as we pointed out earlier is reasonably efficient.

Memory Overhead

Given the memory constraints of sensor nodes, it is crucial that any proposed key management protocol does not A node needs to store two symmetric keys (one key for the link to the network manager, and one network-wide key for calculating the link-layer MAC). A node also needs to temporarily store a second link key to the network manager during the rekeying phase and a group key during the software update procedure. Assuming we use 128-bit symmetric keys, the total storage needed is 64 bytes for the four keys. In addition, the node needs to store the network manager's public key. Using 128-bit ECC, the size of this key is 16 bytes. Thus, the total key storage needed is 80 bytes.

6.4.10 Analysis of Security Properties of the Key Management Protocol

The link key rekeying process provides backward and forward secrecy. On the contrary, the group key distribution does not provide backward secrecy. There is a trade-off between security and efficiency of the rekeying protocol. In this work, we have chosen an efficient method for distributing a key generated in the network manager over a method to synchronize and generate the same key on a number of nodes. Keeping the nodes in the group synchronized is computationally expensive. For example, MiniSec [110] avoids counter synchronization for broadcast communication since it is too expensive for many receivers. If a node is physically compromised, the link key can be extracted and used to decrypt earlier recorded messages to get the group key. The attacker can then use the group key to decrypt the distributed software binary. However, if an attacker has physical access to a node, he has already gained access to the software running on the mote. Therefore, providing backward secrecy for the group key by using a different protocol is too costly without having any benefits for additional security.

However, we can achieve a certain level of backward secrecy using the current protocols; the link key can be rekeyed immediately after the group key has been established. If an attacker compromises a node physically after the group key has been deleted, the attacker is prevented from reconstructing the group key from the current link key and previously recorded messages. This provides a certain level of backward secrecy. However, there is still a time window where both link keys as well as the group key are stored on the node. If the attacker compromises the node during this time, he can access the software on the node, decrypt the software binary using the stored group key, and reconstruct the group key from the old link key and recorded messages.

If the private key of the network manager is compromised, an attacker can impersonate the network manager. To reduce the risk of successful cryptanalysis of the private key, the private/public key pair of the network manager can be rekeyed.

6.5 Discussion

Wireless sensor networks are becoming more prevalent in process control system applications, such as SCADA systems. Given the impact of attacks on the SCADA system on the nation and overall economy, it is crucial that the integration of sensor networks into the existing SCADA infrastructure does not increase the security threat of these systems. In this chapter, we focused on two problems: 1) intrusion detection, and 2) key management and secure software update.

We proposed a multi layer model-based intrusion detection system for SCADA applications. Our proposed solution is a combination of central and distributed IDS agents that work together to detect anomalous behavior among sensor nodes. We described normal behavior at various levels of network stack, such as the physical layer, link layer, and routing layer. Our solution also suggests possible responses to detected intrusions based on the intrusion threat level.

We also proposed a link-layer rekeying protocol and methods for secure software update. An issue that requires further consideration is rekeying of the network-wide key in a synchronized and secure manner. The rekeying procedure must also be efficient and prevent compromised nodes from getting the new network key. A protocol for rekeying of the private/public key pair of the network manager must also be designed. An initial approach is to sign the new public key with the old private key such that nodes can verify the authenticity of the new public key (PK_{NM2}) using the old public key (PK_{NM1}). Once the new key is verified, the old key can be deleted. The protocol must be efficient and ensure that all parties have received the new public key.

Part of the future research includes implementing our proposed solutions (model-based IDS, key management protocol, and secure software updates) on a SCADA test-bed. Given the difficulty and cost of building a real SCADA test-bed, we plan to build a simulation test-bed. The initial steps toward developing this test-bed have been taken, and the development

process is underway. For further information on the design of the SCADA test-bed and the current status of the implementation, the reader is referred to [121].

Chapter 7

Health-care System

The new health care solutions utilize ad-hoc networking principles and wireless communication technology of sensor networks to provide continuous and non-intrusive health monitoring regardless of a patient's or a care-giver's location and activity. In addition, sensors are used in the hospital setting to allow doctors, nurses and other care-givers to continuously monitor the vital signs and status of their patients. Given the time critical nature of health care interventions, Wireless Sensor Networks (WSN) in medical applications must have well-defined reliability and trustworthiness metrics. There are a number of issues regarding the laws and policies surrounding privacy of health care information, including: data access and storage, data mining, and conflicting regulatory framework. More details can be found in [6]. However, the main focus of the research presented in this chapter is to develop tools to defend the health care system against malicious intrusions, such as the insertion of malicious data, which would result in compromising the integrity of the entire system and the privacy of its users. In fact, an important aspect of preserving integrity relates to the system itself rather than only to data items[1][2].

[1] Permission to use parts of these works for the purposes of this dissertation have been granted by the IEEE and/or the appropriate copyright holder(s).

[2] Part of the contents of this chapter are based on [122].

The design requirements for a medical sensor network depend greatly on the specific application and deployment environment. In [123] the authors identify several characteristics that are shared by almost all medical sensor networks. These include: wearable sensor platforms, reliable communications, multiple receivers, device mobility, device lifetime and security.

Sensor networks pose unique challenges in terms of designing security mechanisms, specifically because of power, computation and communication constraints of the individual sensors. As a result, security techniques used in traditional networks cannot be applied directly to the sensor networks paradigm. Moreover, given the unattended nature of the sensor network and the broadcast communication medium, the security threat is much higher than in traditional networks. There are many research efforts in securing wireless networks for health care systems. However, many of these efforts do not consider the issue of data integrity and system robustness. For example, [124] examines the utility of physiological parameters for generating cryptographic parameters in order to secure the communication in a wearable body area sensor network. Our approach is fundamentally different from previous research in that it focuses on designing Intrusion Detection System (IDS) and Integrity Monitoring System (IMS) specifically for WSN in health care system.

The main contributions of this chapter are: characterizing the types of data alteration, proposing metrics for determining the severity of attacks on the data integrity and reliability, and proposing architectures for an IDS and an IMS.

The rest of this chapter is organized as follows. Section 7.1 gives the background information on sensor networks in health care systems. Section 7.2 describes the design issues of IDS and IMS for WSN health care systems. Section 7.3 describes the components of our proposed solution, and Section 7.4 explains the details of the integrated IDS and IMS system designed to protect the integrity and privacy of the health care sensor network. Section 7.5 discusses how to build the correlation model for the sensor data gathered through exper-

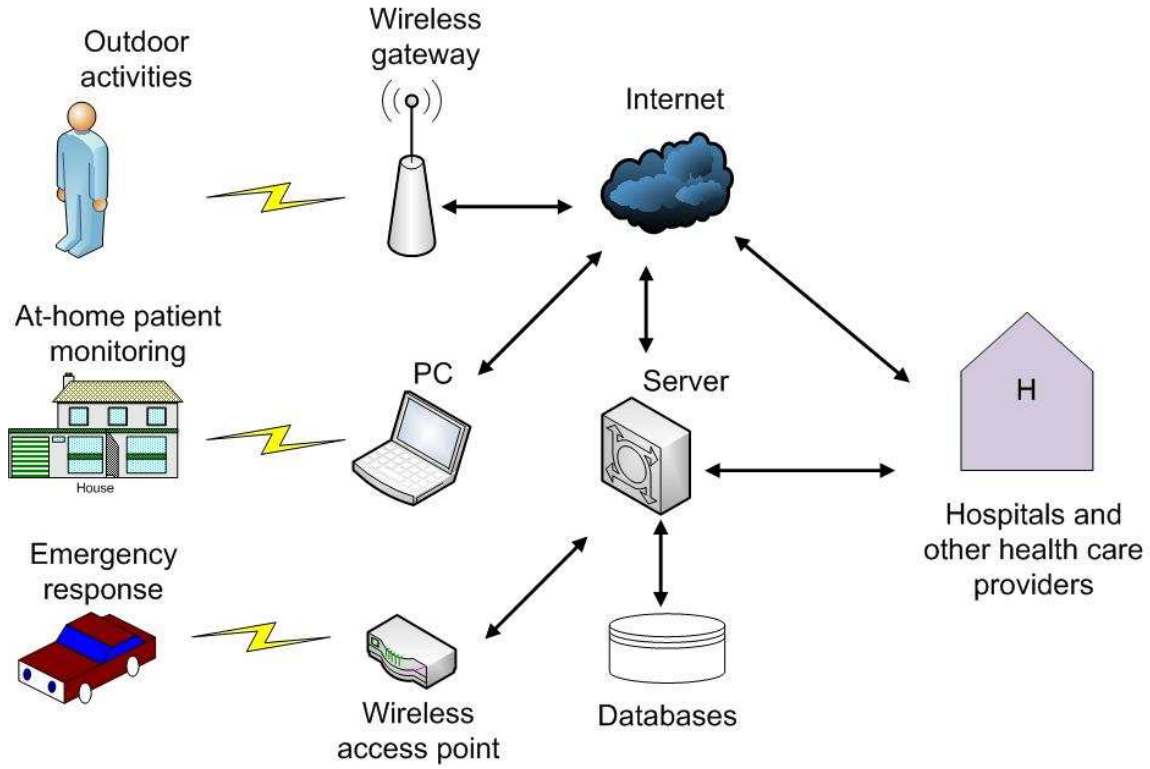


Figure 7.1. A typical architecture for WSN in health care applications, such as at-home patient monitoring [6].

iments. Finally, Section 7.6 discusses the future directions for research in sensor networks used for health care applications.

7.1 WSN in Health Care Systems

The general architecture of the WSN in health care applications is shown in Figure 7.1. The bodily sensors are deployed at the site where the patient is being monitored. These sensors monitor the vitals signs, and send their collected data to a PC or PDA. The PC (PDA) forwards the collected data to the health care provider sites, such as hospitals, through a network (typically Internet).

In the rest of this section, we briefly review some of the existing health care applications that use WSN.

CodeBlue [125] developed at Harvard University is a WSN deployed in emergency medical care and response. The system uses sensors for monitoring vital signs, PDAs, and PCs for monitoring and responding to the patients needs. The goal of CodeBlue is to efficiently transfer data so that the health care professionals and first respondents can diagnose and allocate resources for the patient.

Scalable Medical Alert and Response Technology (SMART) [126] is developed by Brigham and Women's Hospital. SMART is a WSN system for tracking and monitoring patients at home, and continues the service through transfer, triage, hospitals or other health care facilities. Patients, doctors, nurses, and equipments are located by the SMART system on demand through the use of a location-aware service.

MobiHealth [127] provides value-added services based on the 2.5G and 3G technologies. The system integrates sensors and actuators to monitor vital signs and transmits these readings to the health care provider, such as a hospital. The readings are in the form of audio and video feeds. These services are used for disease prevention and diagnostics, remote assistance, para-health services, physical state monitoring, and clinical research.

Other examples of WSN in health care applications include, CustoMed [128] and the deployed system proposed in [129].

7.2 Intrusion Detection System for Health Care WSN

The goal of this work is to ensure system robustness, which we define it to be: assuring that the system is performing well (availability, reliability and accountability), and all of the system components, including any software and/or hardware, are not faulty.

Traditionally, cryptography has been used as the first line of defense to ensure data integrity. However, cryptographic schemes works under the assumption that an attacker is not able to gain access to hardware. This assumption fails in the sensor network paradigm since

these networks are generally deployed and left unattended. If an adversary captures a sensor, he can easily extract the cryptographic primitives and keys as well as exploit the shortcomings of the software implementation [130]. Once the adversary has the cryptographic key, he is able to access the sensory data in order to modify or to exploit it. This motivates the need for an IDS, i.e. an application-level module that is able to detect and to take the right countermeasures against attacks.

Generally, data can be altered or corrupted in two ways:

- *Maliciously*: when an attacker, who has accessed the cryptographic keys, modifies the information in the system.
- *Incidentally*: when a hardware malfunction result in incorrect data.

Regardless of the cause of the corruption, the IDS must be able to detect abnormal data. A distinct characteristics of wireless sensor networks in health care applications is that the entire system is composed of completely orthogonal entities (that have dynamic behavior over time): sensor nodes, people and the network used to transmit data. We must consider each of these components, their behavior, and the possible threats posed by each of them in order to best design techniques to ensure integrity and reliability of the entire system.

7.3 Proposed Components

In this section we discuss the four main components of our system and conclude with ways to integrate them.

7.3.1 Data Classification

An intrusion detection system (IDS) is used to detect malicious behaviors that can disrupts the functionality and security of a system, such as a computer network. The IDS achieves this

task by categorizing the data into *good* and *bad* or *malicious* data. This characterization of the data is an essential parameter in designing a successful IDS. Once the models for acceptable and unacceptable traffic have been built, the incoming observations will be compared to these models and classified as acceptable or anomalous. When suspicious observations are detected, an alert is triggered. For example, in the context of health care applications, the detection of anomalous data should prevent drug delivering actuators from engaging and request prompt investigation by a member of the medical staff. Possible models to classify the incoming data are:

Value range: For each sensor output the highest and the lowest possible values are registered and if the incoming data are outside this range, the data is considered malicious. This very simple model protects the system against naíve attackers that do not have any knowledge of the meaning of the data they are corrupting.

System dynamics: Values of each observation usually follow certain patterns. Therefore, some behaviors in time can not be possible. For example, a person's heart beats 70 times in a minute, and that is the pattern one expects to see when monitoring a patient. Therefore, if we observe a sporadic pattern of heart beat that does not follow the pattern we are used to, this might raise a flag in the system. The model of how each observation value evolves in time can be built by considering the spatial and temporal behavior of the monitored object using the gathered data. It is worth mentioning that depending on the application of the health care system, the model of acceptable behavior in time might change. For example, if the system is monitoring a cardiovascular patient, then a sporadic heart beat might signal a heart attack instead of a malicious attack. Therefore, the model for behavior in time is dependent on its application.

Correlation model: Observations from different sensors are not completely independent of one another. For example, heart rate and activity level of a person are highly correlated. We need to model this correlation among various observed signals so that the system is capable of

determining unacceptable correlations and triggering an alert if an anomaly is detected. These types of models are the most effective to defend the network against the most skilled attacker. H/she can modify the range in a way that models based on behaviors do not trigger any alerts. This is a less probable that an attacker has the knowledge and the power to modify all the sensor readings to maintain their correlations acceptable. The implementation challenge here is to identify the interdependence among sensor readings. This requires support from medical doctors that know how sensory observations correlate.

7.3.2 System Criticality Metric

WSN for health care applications can be categorized according to their criticality. This does not depend on the network itself, but exclusively on how the network is used. For example, the use of sensors to monitor leisure physical activities is less critical than the use of sensor devices for heart rate monitoring of patients in a cardiovascular center. In the latter case, threats to integrity and reliability of the system translate into serious consequences to the patient's health. If sensors are connected to actuators, the result can be quite harmful.

Some of the dimensions that must be considered are: threat to the life or health of a person, loss of an individual's reputation, financial hardship, number of people involved, and the consequences of an action. The complete analysis of all the characteristics of the system implementation will require a long time and a great deal of effort.

7.3.3 Relevance of Observations

Depending on the application at hand, the observations have *high*, *moderate*, or *low* relevance to the outcomes of the system. This relevance is closely related to the sensitivity of each observation. Sensitivity, in our context, is defined as the magnitude of the change to the status of the system given the added observation.

While criticality is a characteristic of the entire application, observation relevance is a conditional characteristics, i.e. characteristic of each output given the particular application.

7.3.4 Robust Inference

The robust inference engine comprises the final component of an IDS. The task of this engine is to make recommendations to the system administrator and actuators so that these entities are capable of making proper decisions. Once the traffic anomalies are detected and identified as malicious tampering or hardware malfunction, the robust inference engine uses these results to provide suggestions to the system.

7.4 The Integrated System

In order to implement our described system, we need algorithms that are capable of tracking anomalies and intrusions based on streams of observations. Each possible scenario is a complex processes that can be detected and tracked. The large volume of data collected by WSN is a combination of informative data and background noise from the environment or sensor hardware. Powerful correlation engines must be used to detect activities given noisy observations.

In order to describe the correlation engines, we must first describe activities to be detected through Hidden Discrete Event System Models (HDESM). HDESM are discrete event dynamical system models whose underlying internal state space is not directly observable. The distribution of an observation of a HDESM is typically given by a probability distribution conditioned on the hidden state of the system. Two computational problems arise in this framework. The first problem is determining the most likely process-to-observation association, called the Discrete Source Separation Problem. The second problem is concerned

with determining the ”best” assignment of events to process instances given a sequence of observations and an HDESM process.

Our challenge is to find solutions to the above mentioned problems. These problems are not solved by traditional information retrieval and database query approaches. In our system design, we plan to use the Markov Chain Monte Carlo Data Association (MCMCDA) [5] (mentioned in Chapter 5) method and the Process Query System (PQS) Engine [89]. PQS is used to detect processes using the data observations. These powerful tracking techniques compute rated hypothesis of consistent tracks given the observed data. These algorithms have proven to be very successful and robust in many challenging applications [90]–[92], [131].

The MCMCDA methodology is an efficient real-time algorithm that solves the data association problem and is capable of initiating and terminating a varying number of tracks. MCMCDA is suitable for sensor networks since it operates with no or incomplete classification information. Process Query System is a powerful software front-end to a database or a real time sensing infrastructure, that allows users to define processes at a high level of abstraction and submit process definition as queries. Missed detection and disambiguation of multiple processes are handled within the PQS kernel. The system uses a library of hidden state sequence estimation algorithms, such as Viterbi-type algorithm for Hidden Markov Models, and Kalman-type algorithms to evaluate state sequences.

The MCMCDA hypothesis generation is based on both current and past observations whereas PQS is a Multiple Hypothesis Testing (MHT) type of tracker that uses recursive filtering techniques to estimate the current state of each track as a function of the current observation being assigned to it, and the previous estimate. As a consequence, MCMCDA does not need to maintain multiple hypotheses, but it needs to reconsider all the observations in order to compute a new hypothesis. PQS, on the other hand, updates the estimate of each track much more efficiently. However, PQS needs to consider several possible associations of observations to existing tracks which could lead to a potentially geometric growth of the

hypotheses set. Given these differences, a future research direction is to compare the two protocols in terms of their applicability to the health care systems.

7.5 Experiments: Data Modeling

In this section, we describe the initial steps taken toward developing models for health care applications. The data we use in this section were collected (and provided to us) through the experiments performed by researchers at UC Berkeley, Vanderbilt, Cornell, and UT Dallas under 'The TRUST Home Medical Sensing Testbed' project.

In these experiments, patients' movement data was collected using a number of bodily sensors (tmote sky motes [132]) equipped with a custom sensor board developed at UC Berkeley. The sensorboard is composed of a 3-axis accelerometer and two 2-axis gyroscopes which together provide three orthogonal gyroscope measurements, shown in Figure 7.2. These sensors have been used in conjunction with a secondary network of Stargate boards to capture data from the worn devices over a larger area. In addition, these boards captured images of the patients to aid in the system development.

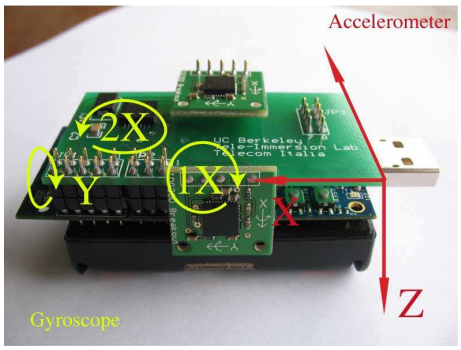


Figure 7.2. Motion sensorboard and tmote sky with sensing axis labeled courtesy of 'The TRUST Home Medical Sensing Testbed' project.

The data we use in our modeling was collected by the following procedure. Five motes were attached to the body of the patient: 1)Left Wrist, 2)Right Wrist, 3)Waist, 4)Left Ankle, and 5)Right Ankle. The data was collected from the patients performing a number of activi-

ties: left hand rising, right hand rising, both hands rising, left leg rising, right leg rising, both legs rising, drinking water, sit to stand and stand to sit, walking, going up and downstairs, and planting. An example of the parsed data is shown in Figure 7.3.

In order to model the correlation among the various parameters for each activity, we use Auto-regression (AR) with a sliding window. AR is a procedure used in time series analysis in which a multi-dimensional vector is transformed into a number of coefficients to make the analysis much easier. AR is used to capture the evolution and interdependencies among time series.

An AR model describes the evolution of a set of k variables measured over the same sample period, i.e. $t = 1, \ldots, T$, as a linear function of only their past evolution. The variables are collected in a $k \times 1$ vector y_t, which has as the ith element $y_{i,t}$, where t is the time of observation of the variable y_i. A p-th order AR, denoted $AR(p)$, has the following form:

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t \tag{7.1}$$

where c is a $k \times 1$ vector of constants (intercept), A_i is a $k \times k$ matrix for every $i = 1, \ldots, p$ and e_t is a $k \times 1$ vector of errors. Intuitively what this procedure does is to decompose a complex signal into smaller signals that can be monitored easily over time. In this case, we find the coefficients A_i and use a sliding window[3] to monitor the changes in these coefficients over time, i.e. we track A_i^t. If there is a sudden change in any of these coefficients over time, then the monitoring program will raise a flag. Coefficients A_i give the correlation among different signals.

For our set of data $p = 5$. We used Matlab to find the autoregression coefficients for each time step, and used a sliding window of size $t = 30$. In order to track the behavior of the A_i's over time, we use the largest eigenvalue of the matrix[4]. Figure 7.4 shows the result of

[3]The sliding window moves one time step at a time.

[4]It should be noted that we can use other matrix norms as well.

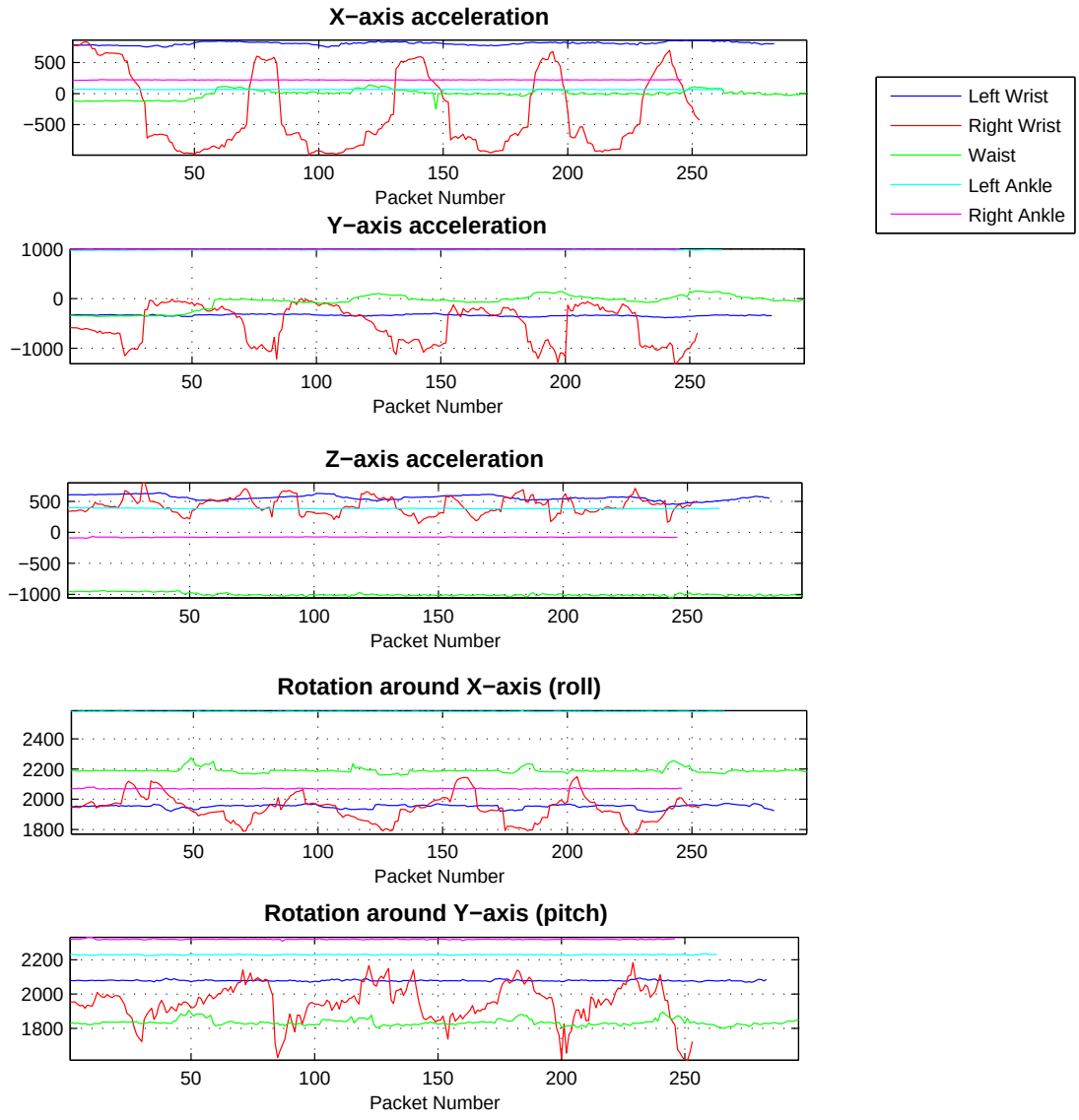


Figure 7.3. In this figure the patient was rising his right hand repetitively.

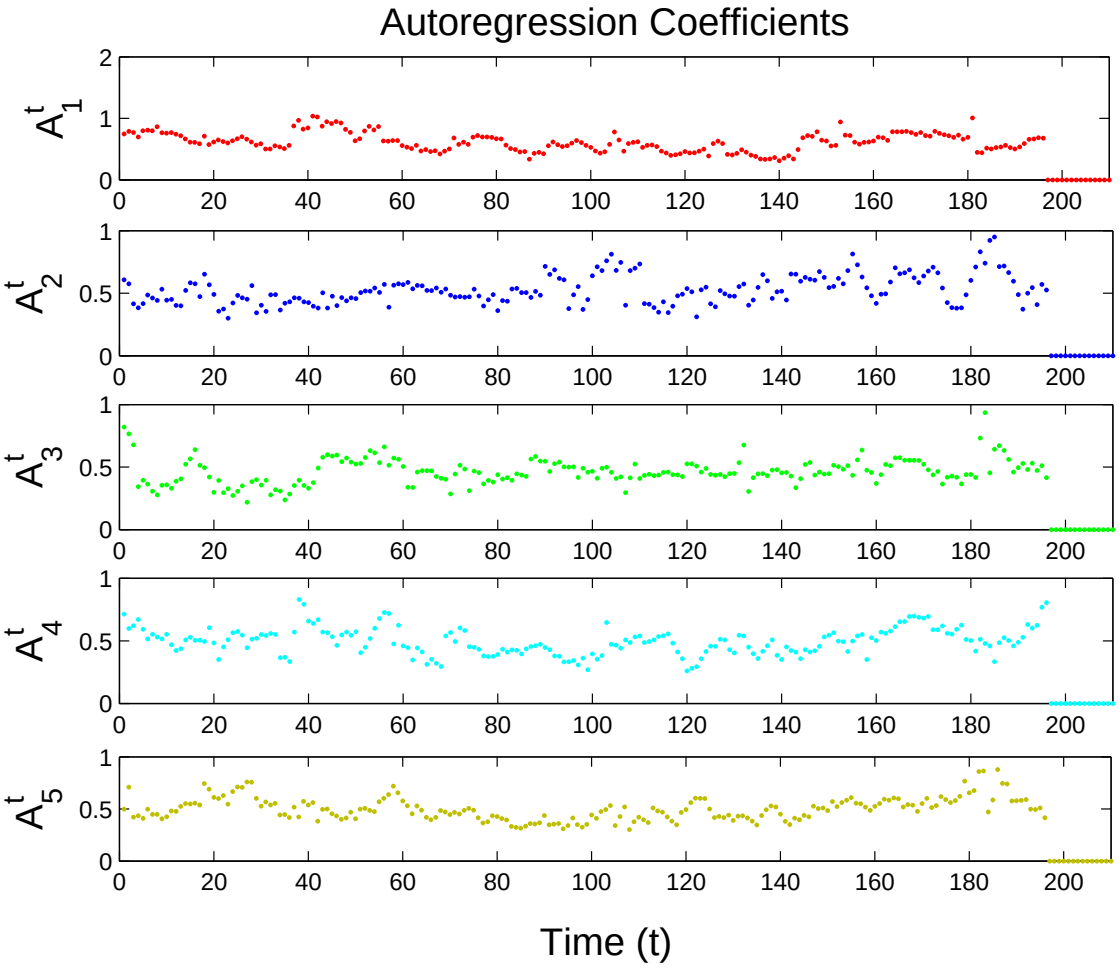


Figure 7.4. Tracking the largest eigenvalue of the autoregression coefficients, A_i, over time using a sliding window of size 30.

tracking the largest eigenvalues of each autoregression coefficient. As can be seen from the plots, the largest eigenvalues of each coefficient remains within a bounded value over time. This can be used as the model of data correlation that we feed into the PQS module. If the largest eigenvalue of the coefficient matrices spikes at a give time step, it could signal an anomaly.

7.6 Discussion

WSN for health care systems transmit a large volume of data collected from several bodily sensors. Given that the information regarding the health of an individual is highly sensitive and critical, it must be kept private and secure. Consequently, it is important to equip the system with mechanisms that would prevent unauthorized agents from acquiring or tampering sensitive data that is transmitted over the network and stored in dedicated repositories.

In this chapter, we proposed security solutions that included developing intrusion detection and integrity monitoring systems by defining the relevant metrics and describing the components of the system. Next, we described using a sliding window autoregressive parameter estimation to model the correlation among the data gathered from the bodily sensors. The largest eigenvalues of the coefficient matrices can be used to detect anomalies. Finally, we discussed MCMCDA and PQS algorithms, which are the two possibilities for developing an Intrusion Detection System to track anomalies and intrusions (using the autoregressive model).

This work is an ongoing research project. As part of the future work, MCMCDA and PQS must be compared in terms of their speed of convergence and accuracy of the generated hypotheses. This comparison will aid in determining which algorithm is a better fit for the health care system application. In addition, there is a need to define a state space model based on the real health care data, which is available to us from a test-bed, that can be used by MCMCDA and PQS.

Part 4

Analytical Tools for Security

Chapter 8

Game Theory

As it has become clear throughout this dissertation, sensor networks are vulnerable to physical node compromise. This problem can not be resolved by solely using cryptographic methods. Therefore, it is important to employ other methods that help determine compromised nodes and avoid using them for performing distributed tasks in sensor networks[1].

A reputation based data forwarding mechanism is one of the powerful non-cryptographic methods. This method has been used in wireless ad-hoc networks, and as shown in Chapter 5 it can be used in sensor networks. It has to be noted that in our context reputation and trust are used synonymously. Therefore, from this point on, we use the term trust to refer to how much confidence a node has in another node to perform the required operations correctly.

A trust mechanism is defined by two essential components: a trust metric (value) and a trust threshold [44], [134]–[137]. Trust metrics may be abstracted in a manner that can be implemented on computers, making them of interest for the study and engineering of distributed systems. A cutoff value called 'trust threshold' is used by the nodes to decide whether they will have future interactions with another node or not. This threshold is related to the maximum number of times that a node can perform its task incorrectly before other

[1]The contents of this chapter are based on [133].

nodes consider it to be untrustworthy. For example, how many times a node does not forward a data packet that is sent to it.

Analytical models to evaluate the performance of sensor networks have been scarce due to the distributed and dynamic nature of such networks. Game theory is conceptually a powerful tool that allows us to make predictions on how strategic players would behave, when all strategies interact with each other to dictate the final outcome. Therefore, game theory offers a suite of tools that may be used effectively in modeling the interaction among independent nodes in a sensor network. In particular, the repeated game framework allows the modeling of a game among independent nodes that over time. The repeated game framework has already been used extensively in analyzing the power control and medium access control (MAC) in wireless ad-hoc networks [138], [139]. However, this framework has not been adopted for the analysis of trust systems in wireless sensor networks. In this chapter, we define a repeated game that models the trust system in sensor network and solve the game to arrive at strategies that can be used in a sensor network application.

The rest of the chapter is organized as follows. Section 8.1 gives the background on trust systems used in wireless sensor networks. Section 8.2 describes the details of the trust game, such as the game parameters and utility functions. Section 8.3 discusses how the game evolves over time and various stages of the repeated game. Section **??** describes the process of Bayesian learning by the players in the game. Section 8.5 gives proofs of our theorems pertaining to the trust game in sensor networks.

8.1 Trust Systems in Wireless Sensor Networks

There have been a few trust mechanisms that have been proposed for use in wireless ad-hoc and sensor networks, such as [44], [136]. In a trust system a node, called the trustor node, observes the communication of a neighbor node, called the trustee node, after it has

passed a data packet to the trustee node. It is conventional to assume that a node can overhear communications of its neighbor nodes using the promiscuous mode of operation. In fact, the promiscuous mode of operation is used in many in routing protocols that have been developed for wireless ad hoc and sensor networks, such as, Dynamic Source Routing (DSR) protocol [140][2]. The trustor node observes if the trustee node correctly relays the data packet towards the final destination. If the data is correctly forwarded, then the interaction between the node and its neighbor is considered to be successful, otherwise it is considered unsuccessful.

8.2 General Trust Game in Sensor Networks

In order to model the interactions between a network of trustor nodes and a trustee node, we adopt the framework of the Iterated Heterogenous Trust Game (IHTG) proposed by Buskens in [141]. Buskens uses IHTG to model human interactions in social networks. The analysis in [141] can be carried over to the realm of sensor networks. However, fundamental changes have to be made to the IHTG to model the concepts of trust systems for sensor networks. For example, we need to include information asymmetries and Bayesian-learning into the IHTG framework. In the following sections we present our extended IHTG model, suitable for sensor networks, along with the solution concepts pertinent to this game.

8.2.1 General Games

The essential elements defining any game are: 1)a set of players, 2)the set of actions for each player, 3)a utility function defined over the product set of all the action sets of all players, 4) the information known by any player prior to choosing an action, and 5) the set of rules for playing the game. It is also important to define the duration of the game, meaning if

[2]These observations could be done at the data packet level or at the message level, but we do not make any distinction between these two cases in our analysis.

the game is a one shot, a finitely repeated, or an infinitely repeated game. In order to be able to make any statement regarding the game, we need to have a valid solution concept. Valid solution concepts for the game are equilibria, or system fixed points, in which the system is in a state of balance between opposing forces or actions. In most cases, the solution concepts are equilibria that define the best response strategies for every player in the game. In other words, equilibria are strategies such that no player receives a higher utility by unilaterally deviating and using another strategy [142].

8.2.2 Trust Game Overview

In this section, we are going to describe the repeated game model that we use in our trust game. This game is described by $\Gamma(F_\Theta, \pi, \omega, \delta, \mathbf{B})$. The repeated game is composed of a single stage game, called γ. The game is shown pictorially in Figure 8.1. The figure shows the game in both the extensive and normal forms.

In order to proceed with modeling the game, we make a set of assumptions that are outlined below:

- We examine a subnetwork of the complete sensor network
- This subnetwork is composed of n trustor sensor nodes and one trustee sensor node.
- The n trustor sensor nodes form a network and establish trust relationships among themselves prior to the start of the game.
- The set of trustor nodes do not have an assessment of the trustworthiness of the trustee node at the start of the game.
- The trustor nodes forward data packets to the trustee and observe whether or not the trustee forwards the packet correctly.
- The trustor nodes always have a packet to forward.

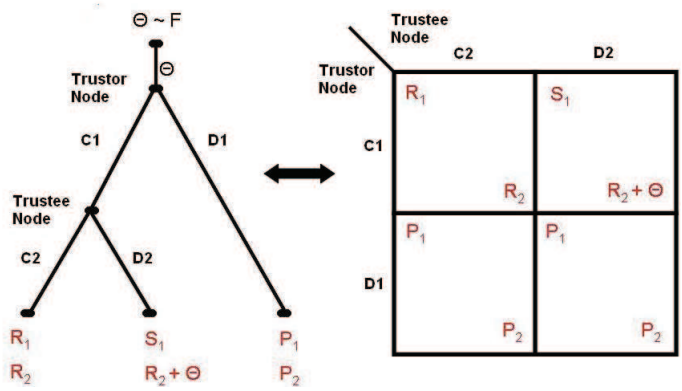


Figure 8.1. One Stage of the Trust Game γ_F; $R_i > P_i$, $i = 1$ for the trustor and $i = 2$ for the trustee, $P_1 > S_1$, $\Theta \in (0, 1]$

- The stage game γ is played repeatedly between one of the n trustor nodes and the trustee node to form the the repeated game Γ. Each stage game represents the opportunity of a trustor sensor node to pass a data packet to the trustee.

The set of actions for each player in the stage game γ is shown in Figure 8.1. The action set of the trustor node consists of C_1 and D_1, where C_1 corresponds to the trustor node forwarding a packet to the trustee node, and D_1 corresponds to the trustor node not forwarding the packet. The action set of the trustee node includes two options, C_2 and D_2. C_2 corresponds to forwarding the packet passed by the trustor node correctly, and D_2 corresponds to not forwarding the data packet correctly.

8.2.3 Game Parameters

We denote parameters associated with the i^{th} trustor node with the index $i1$ and parameters associated with the trustee node with the index 2. We use the index τ to represent the index of the current stage game in Γ starting from stage game 0. This can also be thought of as a time index in the game. We use the index t to represent the number of observations made by a trustor node of the trustee node's actions over time. We use $i1t$ to denote the i^{th} trustor

node's value for t. With these notations, the parameters of the repeated game Γ are defined as follows:

- F_Θ is the cumulative distribution function (CDF) of the random variable Θ, i.e. $F_\Theta(x) = Pr(\theta \leq x)$. F_Θ has full support on $(0, 1]$. Probability density function (PDF) of the random variable Θ is uniform on $(0, 1]$. The meaning of Θ will be explained when we discuss the utilities and payoffs of the game.

- We denote the relative importance of each of the n trustor nodes in the sensor network by $\pi = \{\pi_{\mathbf{1}}, \pi_{\mathbf{2}}, ..., \pi_{\mathbf{n}}\}$[3]. π increases the probability that more important trustor nodes in the network will play γ with the trustee node.

- $0 < \delta_i < 1$ is the dropout rate of sensor i. This rate is dependent on the rate of energy dissipation for the sensor. δ_i is the probability that trustor node i stops interacting with the trustee node in the current stage of the repeated game. However, this does not mean that the trustor i is not going to interact with the trustee again at some time in the future.

- ω is the discount factor, with $0 < \omega < 1$, which is the time-based utility payoff preferences of the n trustor nodes and the trustee node.

- **A** represents an $(n \times n)$ matrix of the probabilities of the information recall (history) given that trustor node i tries to communicate this history of the game to trustor node j. The entry $a_{i,j}$ of the matrix represents this probability and has the property that $0 \leq a_{i,j} \leq 1$.

As noted earlier, the trusters in our game are sensor nodes. A convenient way to represent this network of trustor sensor nodes is to use a graph $G = (V; E)$ where V is the set of vertices (trustor sensor nodes) and E the set of edges (communication links among sensors). Note that we have assumed a connected graph with bi-directional links, i.e. there exists an edge in E between vertices (sensor nodes) i and j only if they can communicate with each

[3]We must have that $\sum_{i=1}^{n} \pi_i = 1$.

other directly. Therefore the adjacency matrix of this graph, denoted by **M**, is symmetric, meaning:

$$\mathbf{M}_{i,j} = \begin{cases} 1 & if(i,j) \in E \\ 0 & otherwise \end{cases} \tag{8.1}$$

The assumption of bi-directional links might not hold true in a real wireless network. However, we use this to simplify the analysis of the game.

We define a second matrix **B** which is defined as the Hadamard product between the adjacency matrix, **M**, and the matrix of probabilities of information recall between each pair of these sensor nodes, **A**:

$$b_{i,j} = (\mathbf{A} \bullet \mathbf{M})_{i,j} = a_{i,j} m_{i,j} \tag{8.2}$$

B will be used to define typical game play and what occurs between successive stages of the game γ.

8.2.4 Utility Functions and Payoffs

In this section, we define the utility and payoff functions of both players, which are the network of trusters and the trustee. Let u_{i1} represent the utility for trustor node i an u_2 represent the utility for the trustee node. In each stage game, γ indexed by τ, we have $u_{i1}(C_1, C_2) = R_1$, $u_{i1}(C_1, D_2) = S_1$, $u_{i1}(D_1, C_2) = P_1$, $u_{i1}(D_1, D_2) = P_1$, $u_2(C_1, C_2) = R_2$, $u_2(C_1, D_2) = R_2 + \theta$, $u_2(D_1, C_2) = P_2$, $u_2(D_1, D_2) = P_2$. This can be seen from Figure 8.1. Without loss of generality, we can assume that all of these utilities are bounded in $[-1, 1]$, $R_1 > P_1$, $R_2 > P_2$, and $P_1 > S_1$. These assumptions are can be justified through the following observations,

- $R_1 > P_1$ since the trustor node associates a higher utility with a successful packet delivery, as opposed to no delivery at all.
- $R_2 > P_2$ since we assume that the trustee node wants to ingratiate itself with the

network of trustor nodes. Therefore, the trustee values receiving a data packet from a trustor node more than not receiving a data packet at all.

- $P_1 > S_1$ since the trustor node associates a higher utility with not forwarding a packet than forwarding a packet and then having the trustee node not forward the packet onwards correctly.

The random variable Θ in this game represents the value of a data packet for a stage γ of Γ. It is assumed that the sensor nodes can correctly assess the value of the data packets they forward to the trustee. Therefore, a realization θ of Θ during each stage of the game, γ, is interpreted as the added incentive for the trustee node to fault on forwarding a data packet correctly. The value of θ can vary throughout the game from data packet to data packet. There is a probability density function associated with Θ, which is assumed to have a full support on the interval $(0, 1]$. To simplify our analysis we assume Θ is uniform on $(0, 1]$.

During each single stage game γ, only one trustor and the trustee play against each other. If a trustor is not involved in a transaction at a given time, it receives a payoff of 0. Payoffs are discounted exponentially with the discount factor ω, where $0 \leq \omega \leq 1$, for the trustee node and all of the trustor nodes. In the repeated game, Γ, each player tries to maximize its own utility; meaning, the trustor nodes try to maximize $u_{i1} = \sum_{\tau=0}^{\infty} \omega^{\tau} u_{i1\tau}$ and the trustee node tries to maximize $u_2 = \sum_{\tau=0}^{\infty} \omega^{\tau} u_{2\tau}$. The exponential discounting indicates that a higher value is placed on the current payoffs as opposed to the future payoffs.

8.3 Evolution of the Game

As noted earlier in the chapter, the repeated game Γ is an infinitely repeated game consisting of an infinite number of single stage games γ. The trustor nodes sequentially play γ with the trustee node. The trustor nodes play the single stage games γ with the trustee node until they drop out of the game at a rate δ_i, defined previously.

8.3.1 Threshold Dependent Actions

In any stage game, γ, a realization θ of the random variable Θ is generated. As noted earlier, θ models the added incentive that the trustee node will fault on forwarding a data packet that could be passed to it by a trustor in that stage. Let us assume that the trustee node has a set threshold value ϑ_{i2}. It should be noted that the trustee sets a value for $\vartheta_{i2} \in [0, 1]$ for its use in playing a game with trustor node i. This value is not changed throughout the game. The value of ϑ_{i2}, in this game, is similarto the concept of a type of player as found in [142]. In any subgame of γ, where a packet is passed to the trustee node by the trustor node i, a value of the realization θ that falls above this threshold causes the trustee node to play the action of incorrectly forwarding the trustor's packet. On the contrary, the values of θ below this threshold result in the trustee node forwarding the data packet correctly.

Each trustor node, $i \in n$, uses a trust threshold ϑ_{i1t} to determine whether or not to forward a packet to the trustee node. It should be noted that we use the notation $i1t$ to represent an index which keeps track of the number of times a particular trustor node i has forwarded a data packet to the trustee node. If the realization of Θ in the stage game falls below this trust threshold, the trustor node i forwards the data packet to the trustee node. On the other hand, if realization θ is larger than ϑ_{i1t}, the trustor node i will not forward the data packet to the trustee node.

8.3.2 Stage Game Transitions

After a single stage of the game is over, there is a possible transfer of the game from the trustor node that just played in γ to another of the n trustor nodes. This transfer of control probability is governed by the probability of a trustor dropping out of the game, δ_i and the relative importance of the trustor nodes π_i. In the next round of the game, the chosen trustor node plays a stage of the trust game with the trustee node. In the context of sensor networks,

this means that the trustor node has the opportunity to pass a data packet to the trustee node in a given round of the repeated game Γ.

The game transition from trustor node i to trustor node j happens with a certain probability that is given by the transfer matrix $\mathbf{\Pi}$, as defined below:

$$\mathbf{\Pi} = \begin{bmatrix} 1-\delta_1+\delta_1\pi_1 & \delta_1\pi_2 & \dots & \delta_1\pi_n \\ \delta_2\pi_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \delta_{n-1}\pi_n \\ \delta_n\pi_1 & \dots & \delta_n\pi_{n-1} & 1-\delta_n+\delta_n\pi_n \end{bmatrix} \tag{8.3}$$

The rows of this matrix indicate the current trustor node playing γ and the columns indicate the player chosen to play the next round γ.

Due to the recall limitations of the trustor nodes, the probability with which a trustor node i will actually pass history information to another trustor node j (when it drops out of the current game γ) is given by the transition matrix $\mathbf{T}$:

$$\mathbf{T} = \begin{bmatrix} 1-\delta_1+\delta_1\pi_1 & \delta_1\pi_2 b_{12} & \dots & \delta_1\pi_n b_{1n} \\ \delta_2\pi_1 b_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \delta_{n-1}\pi_n b_{n-1,n} \\ \delta_n\pi_1 b_{n1} & \dots & \delta_n\pi_{n-1} b_{n,n-1} & 1-\delta_n+\delta_n\pi_n \end{bmatrix} \tag{8.4}$$

Each entry t_{ij} is the probability that the former iteration of the game, γ, was played with trustor node i and that the game iteration proceeds with trustor node j and the trustee at the next stage. When this handoff happens, the past history of the game that trustor node i has is passed from trustor node i to trustor node j. The matrix $\mathbf{T}$ is not a stochastic transition matrix since the transitions in which information is not communicated between two consecutive trustor nodes are not included in $\mathbf{T}$.

8.3.3 Information Recall

In Γ, the trustee is assumed to have perfect memory and perfect recall. The trustor nodes, however, have perfect memory in the game but they do not necessarily have perfect recall during transitions between the trustor nodes, which could occur when one trustor node drops out of the game and another trustor node transitions into play. We assume that only at the moment that one trustor drops out and another trustor enters the game, can myopic history information of the game be communicated between the old and the new trustor nodes and that all of the information is accurate. In this game, we consider the communicated myopic history information to include only the history of played actions in the game by the trustee. However, any information statistics that are calculated by any trustor node (e.g. learned or estimated probabilities of successful forwarding of data packets by a trustee) is not considered a part of the game history that is communicated during trustor node transitions. Such statistics, which are calculated locally by each trustor node, remain in the memory of the trustor nodes throughout the game. This is essential, since the trustee node is assumed to use different thresholds ϑ_{i2} against different trustor nodes i. As we will see later, when a trustor node ties to estimate the value for ϑ_{i2} being used by the trustee when it plays γ with the trustee, history of the trustee's defection against another trustor is not important.

8.3.4 Information Learning

We define the two different models for the additional information that the trustor nodes and the trustee node use during the play of the game. First is the symmetric case, in which both the trustee and the trustor node know the same information. The second scenario is the asymmetric case, in which the trustee node has more information that the trustor nodes. The first case helps develop a set of best-response strategies against trustee node that is just as powerful (in terms of information known) as the trustor nodes. The second case, on the other hand, forms the basis for examining a powerful opponent model, such as those devel-

oped when designing security mechanisms for sensor networks [8]. It is worth noting that we assume that the adversary has different behaviors when playing against different trustor nodes. This assumption allows us to model a more powerful adversary; one that can change its behavior depending on the trustor node which it is playing against in the network.

The information held by each player of the game can be broken down into two categories:

Symmetric Information: In the symmetric information case, we assume that both the value of θ in each stage game γ and the overall distribution of Θ (the added incentive) is known by the trustor nodes. The trustor nodes know the values of the thresholds ϑ_{i2} $\forall i$ of the trustee node, and the trustee node can determine the threshold ϑ_{i1t} $\forall i$ of the trustor nodes. For this scenario, we will see that the solution does not depend on the learning process (or trust value calculation) used by the sensor nodes.

Asymmetric Information: In the asymmetric scenario, the trustee and trustor do not have the same amount of information. The intention is to model the situation in which the trustee node playing against the network of sensors, which trust each other, has more information than the network. Even though the trustee has more information than the trustor node, both players know the distribution of Θ and can observe the realization of θ. However, the trustor nodes do not know the thresholds ϑ_{i2} $\forall i$ of the trustee node. On the other hand, the trustee node can determine the threshold ϑ_{i1t} $\forall i$ of the trustor nodes. Therefore, the trustor nodes need a learning process to estimate the probability of cooperation by the trustee. Later on in the chapter we explain how this is the same as learning ϑ_{i2} for each of the trustor nodes.

8.4 Learning in The Game

As mentioned earlier, the random variable Θ is uniformly distributed in the interval $(0, 1]$. We can model the actions of the trustee node in a stage game γ can be modeled as Bernoulli trials. Let p represent the fixed probability that the trustee node forwards the received packet

correctly when it plays with a trustor node i, so $1-p$ is the probability that the trustee node forwards the received packet incorrectly. Note that by the assumptions and definitions we have $\vartheta_{i2} = p$.

Every trustor node, using a trust system, tries to estimate the value of p using Bayesian parameter estimation [102] when it plays against the trustee node. This is very similar to the reputation system described in Chapter 5. The fact that the posterior probability of binary events is most accurately represented by the Beta distribution, has been derived in [102] and discussed in Chapter 5. To estimate p, each trustor node uses $Beta(x_{i1t}, y_{i1t})$, which has two parameters x_{i1t} and y_{i1t}. The parameter x_{i1t} represents the total number of trustor node i observations of successful packet forwarding by the trustee node at time t. The parameter y_{i1t} represents the total number of observations of unsuccessful events. Estimating p by the trustor node i leads to the estimation of the trustee's thresholds ϑ_{i2} since $\vartheta_{i2} = p$. Some of the properties of the Beta distribution that we use are listed below:

$$
\begin{aligned}
&f(p) = f(p|\alpha,\beta) = Beta(\alpha,\beta) = \tfrac{p^{\alpha-1}(1-p)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}du} \\
&f(p) = f(p|\alpha,\beta) = \tfrac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}p^{\alpha-1}(1-p)^{\beta-1} \\
&\Gamma(s) = (s-1)! \\
&F(p;\alpha,\beta) = \textstyle\sum_{j=\alpha}^{\alpha+\beta-1} \tfrac{(\alpha+\beta-1)!}{j!(\alpha+\beta-1-j)!}p^j(1-p)^{\alpha+\beta-1-j} \\
&\mathbb{E}(p) = \tfrac{\alpha}{\alpha+\beta} \\
&Var(p) = \tfrac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}
\end{aligned}
\tag{8.5}
$$

The above equations represent the probability density function, the cumulative distributive function, the expected value, and the variance of the Beta distribution with $\alpha = 1$, $\beta = 1$.

Using the 8.5, we calculate the probability of a successful event when trustor i plays with the trustee after having t total observation. Having this probability distribution can give some useful information such as the point estimate of parameter p.

We know,

$$
\begin{aligned}
& f(x_{i1(t+1)} = x_{i1t} + 1|p) = p \\
& f(p|x_{i1t}, y_{i1t}, \alpha, \beta) = \frac{p^{x_{i1t}+\alpha-1}(1-p)^{y_{i1t}+\beta-1}}{\int_0^1 u^{x_{i1t}+\alpha-1}(1-u)^{y_{i1t}+\beta-1}du} \\
& = Beta(x_{i1t} + \alpha, y_{i1t} + \beta)
\end{aligned}
\tag{8.6}
$$

This results in,

$$
\begin{aligned}
& Pr(x_{i1(t+1)} = x_{i1t} + 1|x_{i1t}, y_{i1t}, \alpha, \beta) = \int_0^1 f(x_{i1(t+1)} = x_{i1t} + 1|p) f(p|x_{i1t}, y_{i1t}, \alpha, \beta)dp = \\
& \int_0^1 pBeta(x_{i1t} + \alpha, y_{i1t} + \beta)dp = \frac{x_{i1t}+\alpha}{x_{i1t}+\alpha+y_{i1t}+\beta} \\
& Pr(x_{i1(t+1)} = x_{i1t} + 1|x_{i1t}, y_{i1t}, \alpha, \beta)|_{x_{i1t}->\infty, y_{i1t}->\infty} = \frac{x_{i1t}}{x_{i1t}+y_{i1t}} = \frac{x_{i1t}}{t}
\end{aligned}
\tag{8.7}
$$

Note that as the number of samples or observations, t, goes to ∞, the prior α and β become less influential in predicting the probability of successful interaction. It is well-known that this method of Bayesian prediction for the probability of success of the next observation converges to the maximum likelihood estimate (point estimate) when the prior is uniform, i.e. $\alpha = 1, \beta = 1$.

For our scenario, we use the mean of the posterior of Beta distribution to calculate a point estimate, $\tilde{p_{i1t}}$, of the parameter p:

$$
\tilde{p_{i1t}} = \mathbb{E}(p) = \frac{x_{i1t} + \alpha}{x_{i1t} + y_{i1t} + \alpha + \beta} = \frac{x_{i1t} + \alpha}{i1t + \alpha + \beta} = \frac{x_{i1t} + 1}{t + 2} \tag{8.8}
$$

This is the Bayesian estimator that the trustor node i uses to determine an estimate of $p = \vartheta_{i2}$. Therefore, all the information that is needed to find the estimate, such as, the number of direct observations of the trustee which trustor node i, have to be kept in the memory of the trustor node. However, this information does not introduce a great amount of computational and memory overhead since all the trustor needs is the sufficient statistics, i.e. the number of successful and unsuccessful interactions with the trustee. The estimator described above is also consistent, and the proof can be bound in [143].

8.5 Solution of The Game

When analyzing the repeated game Γ, we are interested in finding the trigger strategies for the game. A trigger strategy is one in which a player cooperates as long as the opponent is cooperating. However, as soon as the opponent defects, the player becomes non-cooperative. Trigger strategies come in many forms, such as, the famous Tit for Tat [144]. In our trust game, we analyze grim triggers in which a trigger occurs after one defection by the opponent. With the trigger strategy, once the trustee does not cooperate, i.e. does not forward the data packet correctly, the trustor node stops playing with the trustee node in the current stage of the game. In the remaining stages of the game Γ, the sequence of trustor nodes playing γ against the trustee node will not trust the trustee node as long as the history information (including the defection of the trustee node) is passed on to the next trustor node as the game continues. It has to be noted that due to the information recall assumptions (information transition probability), it is possible that the history information of the game is not passed to the next trustor playing the game. This will cause the trustor node that enters the game to pass a data packet to the trustee node as long as $\theta \leq \vartheta_{i1t}$. There are two scenarios in which a trustor node i entering the stage game will not forward a data packet to the trustee in the single stage game γ,

1. If the information history of the game (passed to the trustor node) indicates that the trustee has defected.

2. If the value of $\theta \leq \vartheta_{i1t}$ for the current trustor (the incentive of the trustee to defect is lower than the trustor's threshold).

Therefore, our goal is to define the trigger strategies of Γ in the best way possible so that the trustor nodes can set the value of ϑ_{i1t} in a way to foster cooperation as the game transitions.

8.5.1 Main Theorems

In this section, we prove prove theorems for the symmetric and asymmetric information cases for the game $\Gamma(F_\Theta, \omega, \delta, \pi, \mathbf{B})$. As will become evident, the proofs for the symmetric information scenario are stronger than those for the asymmetric information scenario due to the the learning process. Our proofs use the weak convergence properties of the estimator we defined in the previous section. It should be noted that the proofs for the theorems presented in this section use similar approaches the the work done by Buskens in the area of social networks [141]. Now we proceed with the theorems for our trust game played by the nodes in a sensor network.

Theorem 1. *If a vector of trigger strategies is a subgame-perfect equilibrium in* $\Gamma(F_\Theta, \omega, \delta, \pi, \boldsymbol{B})$, $\vartheta_{i1t} = \vartheta_{i2}$ $\forall i$ *and* $\forall t$ *for the symmetric information game.*

Proof. In Γ, a trustor node i will forward a data packet to the trustee (play C_1) if $\theta \leq \vartheta_{i1t}$, and the trustee node will only forward the packet onward correctly (play C_2) if $\theta \leq \vartheta_{i2}$.

(a) First we assume $\vartheta_{i1t} < \vartheta_{i2}$ for at least one trustor node i. Then, trustor node i does not maximize its payoff because:

- $\vartheta_{i2} \geq \theta \geq \vartheta_{i1t}$ occurs with positive probability since the distribution of θ has full support on (0,1].
- The region $\vartheta_{i2} \geq \theta \geq \vartheta_{i1t}$ will be a region where trustor node i plays D_1, but the trustee node is willing to play C_2. Using Figure 8.1, and that $R_1 > P_1$, we see that these set of strategies correspond to a suboptimal payoffs for the trustor node. Therefore, trustor node i has an incentive to increase its threshold value ϑ_{i1t}.
- For any other region besides $\vartheta_{i2} \geq \theta \geq \vartheta_{i1t}$, the trustor optimizes its payoff with the thresholds ϑ_{i1t}.

(b) Now assume $\vartheta_{i1t} > \vartheta_{i2}$ for at least one trustor node i. Then, trustor node i does not maximize its payoff because:

- $\vartheta_{i2} \leq \theta \leq \vartheta_{i1t}$ occurs with positive probability since the distribution of θ has full support on (0,1].
- The region $\vartheta_{i2} \leq \theta \leq \vartheta_{i1}$ will be a region where trustor node i plays D_1, but the trustee node is willing to play C_2. Using Figure 8.1, and the fact that $S_1 < R_1$, we see that these set of strategies correspond to a suboptimal payoffs for the trustor node. Therefore, the trustor node i has an incentive to increase its threshold value ϑ_{i1t}.
- For any other region besides $\vartheta_{i2} \leq \theta \leq \vartheta_{i1t}$, the trustor optimizes its payoff with the thresholds ϑ_{i1t}.

From (a) and (b) we see that having $\vartheta_{i2} = \vartheta_{i1t}$ is the optimal strategy for both the trustor node i and the trustee. This holds for all trustor nodes i and for all stages t of playing the game Γ. □

In the symmetric information game, the thresholds ϑ_{i1t} and ϑ_{i2} of the trustor and trustee are equal, since both players know each other's threshold choices prior to playing the repeated game Γ.

Theorem 2. *The trust game of* $\Gamma(F_\Theta, \omega, \delta, \pi, \boldsymbol{B})$ *has the following property: At least one subgame-perfect equilibrium exists for the trigger strategies,* $\vartheta_{i1t} = \vartheta_{i2} = 0\ \forall i$ *and* $\forall t$ *in both the symmetric and asymmetric information cases.*

Proof. In a one shot game, the optimal strategy for the trustor nodes is not to trust the trustee, and for the trustee is to defect if trust is placed. Using the idea of *backward induction* [142], the one shot sub-game perfect Nash equilibrium also constitutes an equilibrium of the set of repeated game equilibria. This proves the statement, $\vartheta_{i1t} = \vartheta_{i2} = \vartheta_i = 0$ for all trustor nodes i. □

Theorem 2 states that there is a trivial equilibrium in the repeated game Γ in which all trustor nodes always play D_1 (not placing trust) and the trustee node plays D_2 (defect) in every single stage γ of the game.

Before proceeding with the next lemma, we need to define the following matrix which simplifies the proofs:

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{T} & \mathbf{\Pi} - \mathbf{T} \\ \mathbf{0} & \mathbf{\Pi} \end{bmatrix} \tag{8.9}$$

This block matrix represents the probability of various events happening when the game transitions from one trustor node to another trustor node in the game. Intuitively, each block of the matrix Σ has the following meaning,

- $\mathbf{\Sigma_{11}}$: represents the probability that play of the game transitions from a trustor node that has the myopic history of the trustee's defect in the past to a new trustor node and that this information is passed along to the next trustor node.

- $\mathbf{\Sigma_{12}}$: represents the probability that play of the game goes from a trustor node that has the myopic history of the trustee's defect in the past to a new trustor node and that this information is not passed along to the next trustor node.

- $\mathbf{\Sigma_{21}}$: this probability is zero since play of the game never goes from a trustor node that has myopic history of the trustee's defect in the past to a new trustor node and that the new trustor node receives myopic history about the trustee's defect in the past.

- $\mathbf{\Sigma_{22}}$: represents the probability that play of the game goes from a trustor node that does not have myopic history of the trustee's defect in the past to a new trustor node and that this information is passed along to the next trustor node.

Now we are ready to state our next lemma.

Lemma 3. *The infinite sum of an exponentially time discounted square matrix* $\boldsymbol{A}$*, with entries* $|a_{i,j}| < 1$*, eigenvalues* $0 < |\lambda_i| < 1$*, and discount factor* $w \in (0,1)$*, starting from* $\tau = 1$ *is* $(\boldsymbol{I} - w\boldsymbol{A})^{-1} - \boldsymbol{I}$*. To be concrete:*

$$\sum_{\tau=1}^{\infty} w^{\tau}\boldsymbol{A}^{\tau} = \sum_{\tau=0}^{\infty} w^{\tau+1}\boldsymbol{A}^{\tau+1} = w\boldsymbol{A}\sum_{\tau=0}^{\infty}\tau^{\tau}\boldsymbol{A}^{\tau}$$
$$= \sum_{t=0}^{\infty} w^{\tau}\boldsymbol{A}^{\tau} - \boldsymbol{I} = (\boldsymbol{I} - w\boldsymbol{A})^{-1} - \boldsymbol{I}$$

Proof. Let $w\mathbf{A}$ be any square matrix. The sequence $\{\mathbf{S}_n\}_{n\geq 0}$ defined by $\mathbf{S}_0 = \mathbf{I}$, $\mathbf{S}_1 = \mathbf{I} + w\mathbf{A}$, and with $\mathbf{S}_n = \mathbf{I} + w\mathbf{A} + ... + (w\mathbf{A})^n$ for $n \geq 2$ is the geometric series generated by $w\mathbf{A}$. The series converges if the sequence $\{\mathbf{S}_n\}_{n\geq 0}$ converges:

$$\sum_{n=0}^{\infty}(w\mathbf{A})^n = \lim_{n\to\infty}\mathbf{S}_n$$

We first note that the eigenvalues of $w\mathbf{A}$ are less than one since $|\lambda_i| < 1$ and $w \in (0,1)$. From the properties of matrices, the geometric series converges if and only if $|\lambda_i| < 1$ for each eigenvalue of $\mathbf{A}$. This condition holds true for this matrix of interest to us. Therefore, $\lim_{n\to\infty} w\mathbf{A}^n = 0$. We also have $\mathbf{I} - w\mathbf{A}$ is invertible since it is square and $0 < |\lambda_i|$ holds for $\mathbf{A}$. Next, $\mathbf{S}_n = \sum_{\tau=0}^{n}(w\mathbf{A})^{\tau}$ and $\mathbf{S}_{\infty} = \sum_{\tau=0}^{\infty}(w\mathbf{A})^{\tau} = (\mathbf{I} - w\mathbf{A})^{-1}$ □

Next theorem is concerned with subgame-perfect equilibrium of the trust game.

Theorem 4. *In the symmetric information game, defined by* $\Gamma(F_{\Theta}, \omega, \delta, \pi, \boldsymbol{B})$ *and the transition matrix* $\boldsymbol{T}$*, the vector of threshold trigger strategies* $\vartheta_{i2} = \vartheta_{i1t}$*, is a subgame-perfect equilibrium if and only if* $\vartheta_{i2} = \vartheta_{i1t} \leq (R_2 - P_2)e_i'((\boldsymbol{I} - \omega\boldsymbol{T})^{-1} - \boldsymbol{I})F_{\Theta}(\vartheta_{i2})\ \forall i$ *and* $\forall t$ *in the case of having.* e_i *is the* i^{th} *unit vector of length* n *and* $\boldsymbol{I}$ *is an identity matrix of size* n*.*

Proof. This proof of this theorem follows directly from a similar statement in [141]. Given the parameters of our game are minor alterations to the parameters of [141], we refer the interested reader to see the details of the proof in Buskens. □

Lemma 5. *For any ϵ and for a large enough number of observations t by the trustor node i, the value of $p = \vartheta_{i2}$, $\tilde{p_{i1t}} = \mathbb{E}(p) = \frac{x_{i1t}+\alpha}{x_{i1t}+y_{i1t}+\alpha+\beta} = \frac{x_{i1t}+\alpha}{i1t}$ is such that $|\tilde{p_{i1t}} - \vartheta_{i2}| < \epsilon$ in probability or in expectation.*

Proof. As mentioned in the previous section, the estimator $\tilde{p_{i1t}}$ is a consistent estimator for the parameter $p = \vartheta_{i2}$ of the Bernoulli distribution [143]). Therefore, for any ϵ, we have $\lim_{t\to\infty} Pr(|\tilde{p_{i1t}} - \vartheta_{i2}| < \epsilon) = 1$; resulting in, $\tilde{p_{i1t}} \xrightarrow{Pr} p$. This analysis shows that for the large enough number of observations t by the trustor node and a given ϵ, we have $Pr(|\tilde{p_{i1t}} - \vartheta_{i2}| < \epsilon) = 1$. □

Theorem 6. *In the asymmetric information trust game defined by $\Gamma(F_\Theta, \omega, \delta, \pi, \boldsymbol{B})$, transition matrix $\boldsymbol{T}$, and large number of observations t, the vector of threshold trigger strategies is a subgame-perfect equilibrium if and only if $\vartheta_{i2} \approx \vartheta_{i1t} \leq e_i^{'}((\boldsymbol{I} - \omega\boldsymbol{T})^{-1} - \boldsymbol{I})(R_2 F_\Theta(\vartheta_{i2}) F_\Theta(\vartheta_{\tilde{p_{i1t}}}) - P_2(F_\Theta(\vartheta_{i2}) F_\Theta(\vartheta_{\tilde{p_{i1t}}}) + \epsilon_1))$ $\forall i$. Again, e_i is the i^{th} unit vector of length n and $\boldsymbol{I}$ is an $n \times n$ identity matrix.*

Proof. The PDF of the random variable Θ (added incentive of the trustee node to defect) is given by $f_\Theta(\theta)$, which has a full support on $(0, 1]$ with Θ having no probability of taking on the value 0 (by our assumptions). Therefore, the domain and range of this function is the interval $[0, 1]$. This all implies that $f_\Theta(\theta)$ takes its maximum value, $f_\Theta(\hat{\theta})_{max}$ in the interval $[0, 1]$ by the extreme value theorem.

Let us choose $\epsilon << \frac{1}{|f_\Theta(\hat{\theta})_{max}(R_2+\theta+\mathbb{E}(u_2;D_2|y_{t+1}=y_t+1))|}$ and suppose that t (the number of observations by a trustor node) is chosen such that $|\tilde{p_{i1t}} - \vartheta_{i2}| < \epsilon$ holds $\forall i \in n$. We can see that these choices of ϵ and t are possible due to lemma 5.

To proceed with the proof, we need to define the following constants, $\kappa = f_\Theta(\hat{\theta})_{max}\epsilon$ and $\epsilon_1 = f_\Theta(\hat{\theta})_{max}(R_2 + \theta + \mathbb{E}(u_2; D_2|y_{t+1} = y_t + 1))\epsilon$.

The principle of optimality, discussed in [141] and [138], can be used to analyze repeated games. This principle states that an optimal equilibrium path in a repeated game is one in

which a one step deviation at any decision point of the game does not increase the long term payoff for a player making that decision [145]. We will use the principle of optimality to compare two cases: 1)the trustee node defects (does not forward the data packet) in a one stage of the repeated game Γ, and as a result, could potentially be punished by the network of trustor nodes, 2) the trustee node chooses to cooperate (correctly forward the data packet) when the trustor node playing in that stage of the game forwards a data packet to the trustee.

To simplify notation, we define μ which is a vector with entries $\mu_i = (R_2(F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}})) + P_2(1 - F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}}) - \kappa) + \epsilon_1)$. This quantity is the expected utility derived in a single stage game γ of Γ for the trustee. From the definition of the game, the payoff to the trustee node when the strategy profile (C_1, C_2) is played is R_2. This event occurs with probability $F_\Theta(\vartheta_{i2}, \tilde{p_{i1t}})$. This quantity is equal to $Pr(\theta < \vartheta_{i2} \vee \theta < \tilde{p_{i1t}})$ = $Pr(\theta < \vartheta_{i2} \times Pr(\theta < \tilde{p_{i1t}})$ = $F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}})$ since the two events are independent. When the strategy profile (C_1, D_2) is played, the payoff to the trustee node is $R_2 + \theta + \mathbb{E}(u_2; D_2|y_{t+1} = y_t + 1)$. The probability of this event is $Pr(\vartheta_{i2} < \theta < \tilde{p_{i1t}})$ = $F_\Theta(\vartheta_{i2}) - F_\Theta(\tilde{p_{i1t}})$ The payoff to the trustee node when the strategy profile (D_1, C_2) or (D_1, D_2) is played is P_2. The probability of this event happening is $1 - F_\Theta(\vartheta_{i2}F_\Theta(\tilde{p_{i1t}})) - (F_\Theta(\vartheta_{i2}) - F_\Theta(\tilde{p_{i1t}}))$. Using lemma 5, we can simplify the following expression: $(F_\Theta(\vartheta_{i2}) - F_\Theta(\tilde{p_{i1t}})$ to $\int_{\tilde{p_{i1t}}}^{\vartheta_{i2}} f_\Theta(x)dx \leq f_\Theta(\hat{\theta})_{max}|\tilde{p_{i1t}} - \vartheta_{i2}| < \kappa$.

The expected payoff for the trustee node μ can be easily computed using the defined probabilities and payoffs for the corresponding events. Next we need to compare the expected utility for the trustee node playing D_2 at a single decision point, i.e. $\mathbb{E}(u_2; D_2)$, with the expected utility for the trustee node playing C_2, $\mathbb{E}(u_2; C_2)$ following a similar approach as [141].

Playing strategy profile C_1, C_2:

$\mathbb{E}(u_2; C_2) = R_2 + e_i'((\mathbf{I} - \omega\mathbf{\Pi})^{-1} - \mathbf{I})\mu$. After the strategy profile (C_1, C_2) is played in the current stage, the payoff to the trustee node is R_2. The expected payoff of the trustee node

for this scenario is,

$$\begin{aligned}&\mathbb{E}(u_i; C_2) = R_2 + \textstyle\sum_{\tau=1}^{\infty} \omega^\tau e_i' \mathbf{\Pi}^\tau \mu = \\ &R_2 + e_i'((\mathbf{I} - \omega\mathbf{\Pi})^{-1} - \mathbf{I})\mu\end{aligned} \tag{8.10}$$

Playing strategy profile C_1, D_2:

$\mathbb{E}(u_2; D_2) = R_2 + \theta + e_i'((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I})P_2\mathbf{1} + e_i'((\mathbf{I} - \omega\mathbf{\Pi})^{-1} - (\mathbf{I} - \omega\mathbf{T})^{-1})\mu.$

After the strategy profile (C_1, D_2) is played in the current stage, the payoff to the trustee node is $R_2 + \theta$. First, we need to find the following matrix:

$$(\mathbf{I} - \omega\mathbf{\Sigma})^{-1} = \begin{bmatrix} \mathbf{I} - \omega\mathbf{T} & -\omega(\mathbf{\Pi} - \mathbf{T}) \\ \mathbf{0} & \mathbf{I} - \omega\mathbf{\Pi} \end{bmatrix}^{-1} \tag{8.11}$$

Using the standard 2×2 matrix inversion equation, we get:

$$(\mathbf{I} - \omega\mathbf{\Sigma})^{-1} = \begin{bmatrix} (\mathbf{I} - \omega\mathbf{T})^{-1} & (\mathbf{I} - \omega\mathbf{\Pi})^{-1} - (\mathbf{I} - \omega\mathbf{T})^{-1} \\ \mathbf{0} & (\mathbf{I} - \omega\mathbf{\Pi})^{-1} \end{bmatrix} \tag{8.12}$$

If the trustee node defects in the current round, the trustor node will learn about this defection. The block matrix $(\mathbf{I} - \omega\mathbf{\Sigma})^{-1}$ will then be used to calculate the expected utility for the trustee node in this setting. The expected utility becomes:

$$\begin{aligned}&\mathbb{E}(u_2; D_2) = R_2 + \theta + \textstyle\sum_{\tau=1}^{\infty} \omega^\tau [e_i' \quad \mathbf{0}'] \mathbf{\Sigma}^\tau [P_2\mathbf{1} \quad \mu]' = \\ &R_2 + \theta + [e_i' \quad \mathbf{0}']((\mathbf{I} - \omega\mathbf{\Sigma})^{-1} - \mathbf{I})[P_2\mathbf{1} \quad \mu]' = \\ &R_2 + \theta + e_i'((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I})P_2\mathbf{1} + e_i'((\mathbf{I} - \omega\mathbf{\Pi})^{-1} - (\mathbf{I} - \omega\mathbf{T})^{-1})\mu = \\ &R_2 + \theta + e_i'((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I})[P_2\mathbf{1} - \mu] + e_i'((\mathbf{I} - \omega\mathbf{\Pi})^{-1} - \mathbf{I})\mu\end{aligned} \tag{8.13}$$

Now we examine the situations in which the trustee node has no incentive to have a one step deviation from the equilibrium path of cooperation with the trustor nodes in the game after at least t observations of every trustor node. This is exactly what we need for the principle of optimality: to compare the expected utility of the trustee node from defection in one stage of the game with the expected utility of cooperation in all stages of the game. Using

Equations 8.10 and 8.13, we derive the conditions for the cooperation to be more profitable than defection in one step, meaning $\mathbb{E}(u_i; D_2) \leq \mathbb{E}(u_i; C_2)$ has to hold:

$$\vartheta_{i1t} = e_i'((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I})[\mu - P_2\mathbf{1}] \geq \theta \tag{8.14}$$

Simplifying the above inequality leads to:

$$\begin{aligned} &\vartheta_{i1t} = e_i'((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I}) \\ &\times(R_2(F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}})) - P_2(F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}}) + \kappa) + \epsilon_1) \geq \theta \end{aligned} \tag{8.15}$$

□

Note: the parameters κ and ϵ_1 can be made arbitrarily small for t (number of observations for a trustor node) large enough.

8.5.2 Thresholds in the Continuum Limit

Here we are interested in finding the continuum limit of the thresholds in Equation 8.15. In order to proceed with our calculations, we have to make some simplifying assumptions, the most important of which is that the network of sensor trustor nodes is homogenous. In addition, we assume:

- The drop out rate δ_i is the same for every trustor node, i.e. $\delta_i = \delta$ for $\forall i$.
- All trustor nodes have the same relative importance, meaning, $\pi_i = \frac{1}{n}$.
- The probability of one trustor node transmitting history information to another trustor is the same for all trustor nodes, i.e. $b_{i,j} = b$.

With these set of simplifying assumptions, we can now state our main result concerning the continuum limit of the trust threshold.

Theorem 7. *The continuum threshold limit, i.e. as $n \to \infty$, for all trustor nodes in the homogenous network, with $\delta_i = \delta$, $\pi_i = \frac{1}{n}$, and $b_{i,j} = b$, is:*

- *a) Symmetric information case:*

$$\frac{\omega - \omega\delta + b\omega\delta}{1 - (\omega - \omega\delta + b\omega\delta)}(R_2 - P_2)F_\Theta(\vartheta_{i2}),$$

- *b) Asymmetric information case.:*

$$\frac{\omega - \omega\delta + b\omega\delta}{1 - (\omega - \omega\delta + b\omega\delta)}R_2(F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}})) - P_2(F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}}) + \kappa) + \epsilon_1)$$

Proof. Using the homogeneity assumptions,

$$\begin{aligned}
&\textstyle\sum_{j=1}^{n}((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I})_{ij} \\
&= \tfrac{d-o}{d^2+(n-2)od-(n-1)o^2} - 1 \\
&\text{where,} \\
&o = \tfrac{-\omega\delta\alpha}{n} \\
&d = 1 - \omega + \omega\delta - \tfrac{\omega\delta}{n}
\end{aligned} \tag{8.16}$$

where o represents off-diagonal entries of the matrix $((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I})_{ij}$ and d represents the diagonal entries of the matrix.

Taking the limit as $n \to \infty$ of Equation 8.16, we get:

$$\begin{aligned}
&\textstyle\sum_{j=1}^{\infty}((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I})_{ij} \\
&= \tfrac{1-\omega+\omega\delta}{(1-\omega+\omega\delta)^2 - b\omega\delta(1-\omega+\omega\delta)} - 1 \\
&= \tfrac{\omega-\omega\delta+b\omega\delta}{1-(\omega-\omega\delta+b\omega\delta)}
\end{aligned} \tag{8.17}$$

The threshold for the symmetric information scenario was previously derived, but we write it out here for convenience:

$$\vartheta_{i2} = \vartheta_{i1t} \leq (R_2 - P_2)e_i'((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I})F_\Theta(\vartheta_{i2}) \tag{8.18}$$

And the threshold for the asymmetric information scenario is (for t large enough):

$$\begin{aligned}
&\vartheta_{i1t} = e_i'((\mathbf{I} - \omega\mathbf{T})^{-1} - \mathbf{I}) \\
&\times(R_2(F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}})) - P_2(F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}}) + \kappa) + \epsilon_1) \geq \theta
\end{aligned} \tag{8.19}$$

Combining the results of 8.18 with 8.17, we arrive at the following expression for the symmetric information scenario:

$$\vartheta_{i2} = \vartheta_{i1t} \leq \frac{\omega - \omega\delta + b\omega\delta}{1 - (\omega - \omega\delta + b\omega\delta)}(R_2 - P_2)F_\Theta(\vartheta_{i2}) \tag{8.20}$$

Similarly, combining Equations 8.19 and 8.17, gives us the following threshold expression for the asymmetric information scenario:

$$\vartheta_{i1t} \leq \frac{\omega - \omega\delta + b\omega\delta}{1 - (\omega - \omega\delta + b\omega\delta)}(R_2(F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}})) - P_2(F_\Theta(\vartheta_{i2})F_\Theta(\tilde{p_{i1t}}) + \kappa) + \epsilon_1) \tag{8.21}$$

It should be note that the above thresholds hold true as $n \to \infty$. □

8.6 Discussion

In this chapter, we used game theoretic concepts of trust systems to analyze the the problem of routing in sensor networks. We mapped the Iterated Heterogenous Trust Game (IHTG) from [141] to an appropriate game for sensor networks. We defined solutions to the game through the use of the trigger strategy concepts. In order to prove our results, we made some simplifying assumptions; however, the general solution of the game illustrates how nodes in a sensor network can set optimal trust thresholds by using Bayesian inference and parameter estimation. This optimal threshold helps determine the players's chances of cooperation in the routing process, and in effect, making intelligent decisions regarding forwarding the data packets to the opponent (or trustee).

Chapter 9

Conclusions and Future Directions

We begin this chapter with a high level summary of the major themes and contributions of this dissertation. However, there remains a great deal of questions for future research. In the second section of this chapter, we describe some of the possible directions for future research.

9.1 Summary and Contributions

Sensor networks have found their way into many critical applications, such as health care, surveillance, and process control systems. Therefore, it is important that security issues pertaining to these networks be analyzed. In this dissertation we explored four related themes with respect to sensor networks security:

- We developed a taxonomy of security attacks and existing countermeasures for sensor networks. Although this taxonomy serves as a reference for security attacks, it points out a lack of a holistic view of the overall *security requirements* and *threat models* in sensor networks. Without these notions we cannot evaluate the tradeoffs between resource constraints and security. We also explored the development of methodologies

for evaluation and design of secure sensor network security by defining: (a) security properties and security metrics to help us understand the value of each security solution, (b) a realistic threat model to understand the practical nature of the adversary model in sensor networks, (c) a security design space to identify best practices for the design and configuration of secure sensor networks. This framework can be used to formally define and analyze security attacks and the effectiveness of solutions for each attack and to identify the *path of least resistance* for an attacker. By knowing the path of least resistance, the network designer can develop techniques for allocating resources more efficiently.

- Our second theme explored the issue of insider attacks on fundamental services and applications in sensor networks. This type of attack has a more serious impact on the network since the attacker is in possession of the cryptographic keys and can participate in legitimate inter-node communication. We specifically considered the time synchronization service and the object tracking algorithms. Time synchronization protocols provide a mechanism for synchronizing the local clocks of the nodes in a sensor network. Many applications, such as networking protocols, rely heavily on accurate timing to perform their tasks. We analyze attacks on different categories of time synchronization protocols, show how these attacks affect different classes of protocols, and propose solutions for each attack. We also implement our attacks and countermeasure for one class of time synchronization protocols to validate our analysis. We also analyze the effect of insider attack on multiple object tracking by focusing on a hierarchical target tracking algorithm specifically designed for sensor networks. We develop a hierarchical reputation system framework that helps detect node misbehavior and isolate malicious entities. We evaluate our reputation system experimentally and demonstrate how it improves object tracking in the presence of malicious nodes.

- The third theme in this dissertation deals with the security issues facing the applications

that use sensor networks. We look at two specific that use sensor networks: health care systems, and the process control systems. We develop an integrity monitoring system for the health care application. Next, we develop two security solutions for process control systems: 1) a model-based intrusion detection system, and 2) secure key management and software update.

- Finally, we use a game theoretic framework to analyze and build a distributed reputation and trust mechanism for sensor networks. Game theory provides a way of mathematically formalizing the decision-making process of policy establishment and execution. Game theory has been studied and applied in many fields and applications. However, there has been very limited research in the area of sensor network security. Game theory provides a natural framework for dealing with adversarial situations. Therefore, the object of our research is to analyze the available game theoretic approaches for reputation systems and apply those to field of sensor networks. We developed a trust game for wireless sensor networks in which a set of nodes play against an outside node. The solution to our trust game gives a procedure for dynamically updating the trust value of each node after interacting with the outside node. In addition, the solution to the game results in threshold values that the nodes can use to decide whether they should interact with the outside node or not.

The taxonomy of security attacks in sensor networks, along with some of the countermeasures, was discussed in detail in Chapter 2. Chapter 3 dealt with developing a systematic approach to sensor network security. Instead of categorizing attacks, we classified and ranked the severity of each attack based on the security objective (confidentiality, integrity, availability) it is targeting. Chapters 4 and 5 dealt with the problem of insider attack on some fundamental services in sensor network, i.e. time synchronization and multi-object tracking. In Chapter 4 we discussed the impact of attacks on time synchronization protocols on the applications that rely on correct time for their function. Then we discussed various ways in

which different classes of time synchronization protocols in sensor networks can be attacked. Finally, we carried out experiments on a real sensor network test-bed to show the affect of an insider attack on the Flooding Time Synchronization Protocol. We also proposed a new structure to filter out the bad time synchronization updates, and showed through our experiments that the new structure helps alleviate the problem of bad time synchronization updates.

Chapters 6 and 7 discussed the third theme of security of sensor network applications. We discussed two important applications of sensor networks: 1) health care systems, and 2) process control systems. Chapter 6 discussed the design of a model-based intrusion detection system for the SCADA systems. The contribution of this chapter is to design a model-based intrusion detection system for the SCADA systems. In this chapter, we also proposed a key management and secure software update protocol for wireless SCADA systems. In Chapter 7, we described our integrity-monitoring system for the sensor networks in health care applications. Given the importance of the health care applications, and the risks involved, it is important to ensure that the data collected from sensor nodes are correct, and have not been corrupted by a malicious agent or a hardware malfunction. We developed the first steps toward building a correlation model for the data collected by various bodily sensors. We used auto-regression methods for time-series to analyze the the data and arrive at a set of coefficients that can be monitored over time for fault detection. Finally, in Chapter 8, we defined a repeated game that modeled the trust system in sensor network and proved theorems pertaining to the solution of the game. Even though game theory has been used extensively in economics to model the situations in which opposing sides with conflicting interests play against each other, there has been very little work in applying game theory to the security in sensor networks. Our contribution is to define a repeated trust game in which each agent dynamically updates its belief (trust value) of the actions of the opposite side and makes decisions regarding whether to continue the play or not based on these trust values.

9.2 Future Research Directions

In the last section we outlined the contributions of this dissertation in the area of sensor network security. The dissertation is composed of four related themes: 1) security paradigm, 2) protocol security, 3) application security, and 4) analytical tools for security. In the following section, we discuss some of the possible future research directions for each of these themes.

9.2.1 Security Paradigm

In Chapter 3, we took the first steps toward developing a holistic approach to sensor network security. These steps included the ranking of the attacks using several metrics. As discussed, these attack orderings can be useful to determine the path of least resistance for an attacker. As part of the future work, one needs to further develop the metrics that get used for attack ranking. In addition, the attack ranking along with the type of the attack (i.e. intermediate or final) can be used to develop an attack tree/graph. The cost metric for each attack can be used as the weight of the edges of the graph. Standard graph search algorithms can then be used to find the path of least resistance for an attacker in a given scenario.

9.2.2 Protocol Security

We discussed security of the tracking and time synchronization protocols in this dissertation. As part of the future work, one needs to look at combining robust estimation methods with robust hypothesis formation methods to arrive at a more secure object tracking protocol. As for the time synchronization protocols, there are several possibilities for attacking these protocols, as discussed in Chapter 4. Therefore, one future direction is to develop secure and robust time synchronization protocols. In addition, one has to run experiments with attacking the existing time synchronization protocols to determine the extent of the damage caused by

each attack. The result of the experiments can clarify what has to be fixed and which countermeasures have to be added to the time synchronization protocols to make them more secure and resilient to insider attacks.

9.2.3 Application Security

We discussed two applications of sensor networks: 1) process control systems, and 2) patient monitoring for health care purposes. In addition to these two, there are many other applications of sensor networks. As part of the future research, each application of sensor networks has to be analyzed carefully in terms of security and privacy concerns. Appropriate security mechanisms, such as the ones discussed in Chapter 3, have to be incorporated in each application to ensure proper functionality and integrity of the data. In addition, privacy concerns have to be taken into account when sensors are used in public places, or when they are used in applications that can give out personal information.

9.2.4 Analytical Tools for Security

We used game theory to model a routing game between sensor nodes and an outsider node, which we think of as the adversary. Similar game theoretic formulations can be used to model other security scenarios. For example, one can formulate a game in which the adversary is trying to maximize the damage incurred on the tracking algorithm. The damage is measures in terms of how far the estimated track is from the ground truth. On the other hand, the network is trying to minimize the damage by using a robust estimation method. By solving this game, one can find the optimal parameters that the network has to use for its estimation.

References

[1] J. Regehr, N. Cooprider, W. Archer, and E. Eide, "Memory safety and untrusted extensions for tinyos," *In submission*, April 2006.

[2] J. Elson and D. Estrin, "Fine-grained network time synchronization using reference broadcast," in *The fifth symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[3] A. Ledeczi, A. Nadas, P. Volgyesi, G. Balogh, B. Kusy, J. Sallai, G. Pap, S. Dora, K. Molnar, M. Maroti, and G. Simon, "Countersniper system for urban warfare," in *ACM Transactions on Sensor Networks*, November 2005.

[4] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler, "Spins: security protocol for sensor networks," *Wireless Networks*, 2002.

[5] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for general multiple-target tracking problems," in *In Proc. of the 43rd IEEE Conference on Decision and Control, Paradise Island, Bahamas*, 2004.

[6] M. Meingast, T. Roosta, and S. Sastry, "Security and privacy issues with health care information," in *The 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society,*, 2006.

[7] Crossbow Technology: www.xbow.com/Products.

[8] T. Roosta, S. P. Shieh, and S. Sastry, "Taxonomy of security attacks in sensor networks and countermeasures," *The First IEEE International Conference on System Integration and Reliability Improvements*, 2006.

[9] S. Pai, M. Meingast, T. Roosta, S. Bermudez, S. Wicker, D. K. Mulliganz, and S. Sastry, "Confdentiality in sensor networks: Transactional information," in *IEEE Security and Privacy Magazine*, 2008.

[10] C. Karlof and D. Wagner, "Secure routing in sensor networks: Attacks and countermeasures," in *Ad Hoc Networks, vol 1, issues 2–3 (Special Issue on Sensor Network Applications and Protocols), pp. 293-315, Elsevier*, September 2003.

[11] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proceedings of International Conference on Distributed Computing Systems*, November 2001.

[12] W. Stallings, "Cryptography and network security principles and practices," in *Pearson Education Inc. 3rd edition*, 2003.

[13] K. Okeya and T. Iwata, "Side channel attacks on message authentication codes," *2nd European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, July 2005.

[14] C. C. Tiu.

[15] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems*, November 2004.

[16] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for network sensors," *ASPLOS 2000*, November 2000.

[17] D. Gay, P. Levis, and D. Culler, "Software design patterns for tinyos," *Proceedings of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'05)*, June 2005.

[18] J. Hill, P. Bounadonna, and D. Culler, "Active message communication for tiny network sensors," *UC Berkeley Technical Report, Berkeley*, January 2001.

[19] M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim, "Remote software-based attestation for wireless sensors," *In Proceedings of the 2nd European Workshop on Security and Privacy in Ad Hoc and Sensor Networks*, July 2005.

[20] P. Levis and D. Culler, "Mate : a virtual machine for tiny networked sensors," *ASPLOS*, October 2002.

[21] A. Wood and J. Stankovic, "Denial of service in sensor networks," *IEEE Computer pages 5462*, Oct. 2002.

[22] A. Wood, J. Stankovic, and S. H. Son, "Jam: A jammed-area mapping service for sensor networks," *In Real-Time Systems Symposium*, 2003.

[23] M. Anand, Z. Ives, and I. Lee, "Quantifying eavesdropping vulnerability in sensor networks," in *Proceedings of the 2nd international workshop on Data management for sensor networks*, 2005.

[24] H. Yih-Chun and D. Johnson, "Wormhole attacks in wireless networks," *IEEE Journal on Selected Areas in Communications (JSAC)*.

[25] R. Poovendran and L. Lazos, "A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks," *in ACM Journal on Wireless Networks*, 2005.

[26] Y. Iyer, S. Gandham, and S. Venkatesan, "A generic transport layer protocol for sensor networks," *Proceedings of 14th IEEE International Conference on Computer Communications and Networks*, October 2005.

[27] T. Roosta and S. Sastry, "Probabilistic geographic routing in ad hoc and sensor networks," *In proc. of International Workshop on Wireless Ad-hoc Networks (IWWAN)*, May 2005.

[28] J. Deng, R. Han, and S. Mishra, "Countermeasures against traffic analysis attacks in wireless sensor networks," *The Third ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2005.

[29] P. Kamat, Y. Zhang, and W. Trappe, "Enhancing source-location privacy in sensor network routing," *Proceedings of the* 25^{th} *IEEE International Conference on Distributed Computing Systems*, 2005.

[30] T. He, B. Blum, J. Stankovic, and T. Abdelzaher, "Aida: Adaptive application-independent data aggregation in wireless sensor networks," *ACM Transactions on Embedded Computing System, Special issue on Dynamically Adaptable Embedded Systems,*, 2004.

[31] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," in *Computer Communications*, vol. 30, 2007.

[32] W. Du, J. Deng, Y. Han, and P. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *In 10th ACM Conference on Computer and Communications Security (CCS03)*, October 2003.

[33] L. Eschenauer and V. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 41–47.

[34] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, 2003.

[35] S. Zhu, S. Setia, and S. Jajodia, "Leap: Efficient security mechanisms for large-scale distributed sensor networks," *In 10th ACM Conference on Computer and Communications Security*, October 2003.

[36] R. Pietro, L. Mancin, and A. Mci, "Efficient and resilient key discovery based on pseudo random key pre-deployment," *International Parallel and Distributed Processing Symposium*, April 2004.

[37] C. D. Laboratory, "http:discovery.csc.ncsu.edusoftwaretinyecc."

[38] X. Du, M. Guizani, Y. Xiao, S. Ci, and H. Chen, "A routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks," in *IEEE International Conference on Communications (ICC)*, 2007.

[39] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Proceedings of the third international symposium on Information processing in sensor networks*, 2004.

[40] a. J. L. B. J. Mundinger, "Analysis of a robust reputation system for self-organized networks," *University of Cambridge,Statistical Laboratory Research Report*, January 2004.

[41] P. Michiardi and R. Molva, "Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," *Conference on Communications and Multimedia Security*, September 2002.

[42] S. Buchegger and J. L. Boudec, "Performance analysis of the confidant protocol: Cooperation of nodes - fairness in dynamic ad-hoc networks," *IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing*, June 2002.

[43] M. Carbone, M. Nielsen, and V. Sassone, "A formal model for trust in dynamic networks," *IEEE International Conference on Software Engineering and Fromal Methods*, 2003.

[44] S. Ganeriwal and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," *ACM Security for Ad-hoc and Sensor Networks*, 2004.

[45] S. Moloney and P. Ginzboorg, "Security for interactions in pervasive networks: Applicability of recommendation systems," *1st European Workshop on Security in Ad-Hoc and Sensor Networks*, 2004.

[46] C. Intanagonwiwat, R. Govindan, , and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, August 2000.

[47] T. Dimitriou and D. Foteinakis, "Secure and efficient in-network processing for sensor networks," *Workshop on Broadband Advanced Sensor Networks(BroadNets)*, October 2004.

[48] D. Wagner, "Resilient aggregation in sensor networks," in *ACM Workshop on Security of Ad Hoc and Sensor Networks*, October 2004.

[49] J. Deng, R. Han, and S. Mishra, "Security support for in-network processing in wireless sensor networks," *First ACM Workshop on the Security of Ad Hoc and Sensor Networks (SASN)*, 2003.

[50] J. Elson and D. Estrin, "Fine-grained network time synchronization using reference broadcast," *The fifth symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[51] S. Ganeriwawal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," *The first ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[52] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding synchronization protocol," *Proc. Of the Second ACM Conference on Embedded Networked Sensor Systems*, November 2004.

[53] A. Cardenas, T. Roosta, and S. Sastry, "Rethinking security properties, threat models, and the design space in sensor networks," in *Submitted to Computer Communication Review (CCR), the publication of the ACM SIGCOM*, 2008.

[54] Hart, "http://www.hartcomm2.org/frontpage/wirelesshart.html," *WirelessHart whitepaper*, 2007.

[55] ISA, "http://isa.org/isasp100," *Wireless Systems for Automation*, 2007. [Online]. Available: http://www.isa.org/isasp100

[56] U. S. G. A. Office, "Critical infrastructure protection. Multiple efforts to secure control systems are under way, but challenges remain," Report to Congressional Requesters, Tech. Rep. GAO-07-1036, 2007 2007.

[57] J. Eisenhauer, P. Donnelly, M. Ellis, and M. O'Brien, *Roadmap to Secure Control Systems in the Energy Sector*. Energetics Incorporated. Sponsored by the U.S. Department of Energy and the U.S. Department of Homeland Security, January 2006.

[58] N. W. Group, "Internet security glossary," http://rfc.net/rfc2828.html, May 2000.

[59] M. Manzo, T. Roosta, and S. Sastry, "Time synchronization attacks in sensor networks," in *SASN '05: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, 2005.

[60] T. Roosta, W.-C. Liao, W.-C. Teng, and S. Sastry, "Testbed implementation of a secure flooding time synchronization protocol," in *IEEE Wireless Communication and Networking Conference (WCNC)*, 2008.

[61] D. Dolev and A. Yao, "On the security of public key protocols," in *Proceedings of the IEEE 22nd Annual Symposium on Foundations of Computer Science*, 1981.

[62] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *ACM Trans. Programming Languages and Systems*, July 1982.

[63] J. R. Douceur, "The sybil attack," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 2002.

[64] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *IEEE Symposium on Security and Privacy*, May 2005.

[65] A. A. Cárdenas, S. Radosavac, and J. S. Baras, "Performance comparison of detection schemes for mac layer misbehavior," in *INFOCOM*, 2007.

[66] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wireless Networks*, 2005.

[67] H. Chan, V. D. Gligor, A. Perrig, and G. Muralidharan, "On the distribution and revocation of cryptographic keys in sensor networks," *IEEE Trans. Dependable Secur. Comput.*, 2005.

[68] K. Römer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, December 2004.

[69] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium Research in Security and Privacy*, 2003.

[70] V. Gupta, M. Wurn, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded internet," *Pervasive and Mobile Computing*, vol. 1, no. 4, pp. 425–445, December 2005.

[71] D. Malan, M. Welsh, and M. Smigh, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Cummunications and Networks SECON*, 2004, pp. 71–80.

[72] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *Proceedings of the Fourth ACM workshop on Security of ad hoc and sensor networks*, 2006, pp. 169–176.

[73] Z. Alliance, "http://www.zigbee.org," *ZigBee Specification*, 2007. [Online]. Available: http://www.zigbee.org

[74] N. Corporation, "www.ntru.com/about/ntru_corp.pdf," 2003.

[75] R. Berber, *Methods of Model Based Process Control*. Springer, 1995, ch. 2.

[76] K. Rmer, "Time synchronization in ad hoc networks," in *Proceedings of MobiHoc*, October 2001.

[77] B. Sundararaman, U. Buy, and A. Kshemkalyani, "Clock synchronization in wireless sensor networks: A survey," in *Ad-Hoc Networks, 3(3): 281-323*, May 2005.

[78] B. Hohlt, L. Doherty, and E. Brewer, "Flexible power scheduling for sensor networks," in *Information Processing in Sensor Networks (IPSN)*, April 2004.

[79] S. Coleri, "Pedamacs: Power efficient and delay aware medium access protocol for sensor networks," Master's thesis, UC. Berkeley, December 2002.

[80] R. E. Kalman, "A new approach to linear filtering and prediction problems," in *Journal of Basic Engineering*, 1960.

[81] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," in *Communications of the ACM*, 1981.

[82] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The Flooding Ttime Synchronization Protocol," in *Proc. Of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.

[83] D. B. Reid, "An algorithm for tracking multiple targets," in *IEEE Transactions On Automatic Control*, 1979.

[84] T. Roosta, M. Meingast, and S. Sastry, "Distributed reputation system for tracking applications in sensor networks," *Proceedings of the International Workshop on Advances in Sensor Networks (IWASN)*, 2006.

[85] Y. Bar-Shalom and T. Fortmann, “Tracking and data association,” in *Mathematics in Science and Engineering*, 1988.

[86] Y. Bar-Shalom and X.-R. Li, “Estimation and tracking: Principles, techniques, and software,” in *Artech Houes*, 1993.

[87] S. Blackman, “Multiple-target tracking with radar applications,” in *Artech House, Norwood, MA*, 1986.

[88] R. Karlsson and F. Gustafsson, “Monte carlo data association for multiple target tracking,” in *In IEE Workshop on Target Tracking, Eindhoven, NL.*, 2001.

[89] G. Cybenko, V. H. Berk, V. Crespi, R. S. Gray, and G. Jiang, “An overview of process query systems,” in *in Proceedings of the SPIE Vol. 5403*, 2003.

[90] A. Giani, “Detection of attacks on cognitive channels,” in *Ph.D Thesis, Thayer School of Engineering, Dartmouth College*, 2006.

[91] I. D. Souza, V. H. Berk, A. Giani, G. Bakos, M. Bates, , and G. V. Cybenko, “Detection of complex cyber attacks,” in *in Proceedings of the SPIE, vol 6201*, 2006.

[92] V. Crespi, W. Chung, and A. B. Jordan, “Decentralized sensing and tracking for UAV scheduling,” in *in Proceedings of the SPIE Vol. 5403*, 2004.

[93] V. Berk, G. Bakos, and R. Morris, “Designing a framework for active worm detection on global networks,” in *Proceedings of the IEEE International Workshop on Information Assurance*, Germany, March 2003.

[94] G. Nofsinger, “Tracking based plume detection,” Ph.D. dissertation, Dartmouth College, April 5, 2006.

[95] M. Fan, Y. Tan, and A. Whinston, “Evaluation and design of online cooperative feedback mechanisms for reputation management,” in *IEEE Transactions on Knowledge and Data Engineering*, February 2005.

[96] P. Resnick and R. Zeckhauser, “Trust among strangers in internet transactions: Empirical anlysis of ebay’s reputation system,” in *Advances in Applied Microeconomics: The Economics of the Internet and E-Commerce*, November 2002.

[97] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, “Reputation systems,” *Communications of the ACM,vol. 43(12),pages 45-48*, 2000.

[98] G. Theodorakopoulos and J. S. Baras, “On trust models and trust evaluation metrics for ad hoc networks,” in *IEEE Journal on Selected Areas in Communications*, February 2006.

[99] H. Kuxhner and G. Yin, “Stochastic approximation and recursive algorithms and applications,” in *Springer-Verlag,Second Edition*, 2003.

[100] P. Rousseeuw and A. Leroy, "Robust regression and outlier detection," in *John Wiley and Sons, Inc.*, 1987.

[101] D. C. Hoaglin and F. M. adn John W. Tukey, *Understanding Robust and Exploratory Data Analysis*. John Wiley and Sons, 1983.

[102] A. Josang and R. Ismail, "The beta reputation system," in *Bled Electronic Commerce Conference*, June 2002.

[103] D. Nilsson, T. Roosta, U. Lindqvist, and A. Valdes, "Key management and secure software updates in wireless process control environments," in *ACM Conference on Wireless Network Security (WiSec)*, 2008.

[104] M. F. E.J. Byres and D. Miller, "The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems," in *International Infrastructure Survivability Workshop (IISW), Lisbon, Portugal*, 2004.

[105] "Technical support working group (TSWG). Securing your SCADA and Industrial Control Systems," in *http://www.tswg.gov/tswg/ip/scada.htm*, 2005.

[106] D. Bilar, "Introduction to State of the Art in Intrusion Detection Systems," in *Proceedings of the SPIE International Symposium on Law Enforcement Technologies*, 2000.

[107] A. P. R. da Silva, M. H. T. Martins, B. P. S. Rocha, A. A. F. Loureiro, L. B. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in *Q2SWinet '05: Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*. New York, NY, USA: ACM Press, 2005, pp. 16–23.

[108] "Dust networks, Technical overview of Time Synchronized Mesh Protocol (TSMP). http://www.dustnetworks.com/technology/tsmp.shtml," 2006.

[109] "Key management for scada."

[110] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "Minisec: A secure sensor network communication architecture."

[111] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, "Tinypk: Securing sensor networks with public key technology."

[112] J. Hui, "Deluge 2.0 - tinyos network programming," http://www.cs.berkeley.edu/ jwhui/research/deluge/deluge-manual.pdf, 2005.

[113] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, "Securing the deluge network programming system," in *IPSN '06: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, 2006.

[114] *Cebolla: Pragmatic IP anonymity*, 2002.

[115] I. Hussain and S. M. Mahmud, "Group key management for secure multicasting in remote software upload to future vehicles," Electrical and Computer Engineering Department, Wayne State University, Detroit, MI 48202 USA, Tech. Rep., 2006.

[116] S. Halevi and H. Krawczyk, "Strengthening digital signatures via randomized hashing," May 2005.

[117] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, "Ocb: A block-cipher mode of operation for efficient authenticated encryption," in *ACM Conference on Computer and Communications Security*, 2001, pp. 196–205.

[118] NIST, "Recommendation for block cipher modes of operation," 2001.

[119] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards."

[120] P. Ning, A. Liu, and P. Kampanakis, "Tinyecc: Elliptic curve cryptography for sensor networks," http://discovery.csc.ncsu.edu/software/TinyECC/, 2007.

[121] A. Giani, G. Karsai, T. Roosta, A. Shah, B. Sinopoli, and J. Wiley, "A testbed for secure and robust scada systems," in *RTAS WIP*, 2008.

[122] A. Giani, T. Roosta, and S. Sastry, "Integrity checker for wireless sensor networks in health care applications," in *International Conference on Pervasive Computing Technologies for Healthcare*, 2008.

[123] V. Shnayder, B. Chen, K. Lorincz, T. R. F. Fulford Jones, and M. Welsh, "Sensor networks for medical care," in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005, pp. 314–314.

[124] K.Venkatasubramanian and S.K.S.Gupta, "Security for pervasive health monitoring sensor applications," in *In Proc of 4th International Conference on Intelligent Sensing and Information Processing (ICISIP)*, 2006.

[125] K. Lorincz, D. Malan, T. R. F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, S. Moulton, , and M. Welsh, "Sensor networks for emergency response: Challenges and opportunities," in *In IEEE Pervasive Computing, Special Issue on Pervasive Computing for First Response*, 2004.

[126] D. W. Curtis, E. J. Pino, J. M. Bailey, E. I. Shih, J. Waterman, S. A. Vinterbo, T. O. Stair, J. V. Guttag, R. A. Greenes, and L. Ohno-Machado, "Smart – an integrated, wireless system for monitoring unattended patients," in *To appear in JAMIA*, January 2008.

[127] D. Konstantas, A. V. Halteren, R. Bults, N. Dokovsky, G. Koprinkov, K. Wac, V. Jones, I. Widya, and R. Herzog, "Mobile patient monitoring: The mobihealth system," in *International Congress on Medical and Care Compunetics*, 2004.

[128] R. Jafari, "Medical embedded systems," Ph.D. dissertation, University of California, Los Angeles, 2006.

[129] A. Milenkovic, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," in *Computer Communications (Special issue: Wireless Sensor Networks: Performance, Reliability, Security, and Beyond)*, 2006.

[130] C. Hartung, J. Balasalle, and R. Han, "Node compromise in sensor networks: The need for secure systems," Department of Computer Science University of Colorado at Boulder, Tech. Rep., January 2005.

[131] S. Oh and S. Sastry, "An efficient algorithm for tracking multiple maneuvering targets," in *in Proc. of the IEEE International Conference on Decision and Control (CDC), Seville, Spain*, 2005.

[132] http://www.moteiv.com.

[133] S. Pai, T. Roosta, S. Wicker, and S. Sastry, "Using iterated heterogeneous games to model trust in wireless sensors with limited energies and memories," in *To be submitted*, 2008.

[134] S. Buchegger, "Coping with misbehavior in mobile ad-hoc netoworks," *Thesis*, February 2004.

[135] S. Buchegger and J.-Y. L. Boudec, "A robust reputation system for mobile ad hoc networks," Technical Report IC/2003/50, EPFL-IC-LCA, 2003.

[136] S. Pai, T. Roosta, S. Wicker, and S. Sastry, "Using social network theory towards development of wireless ad hoc network trust," *Proceedings of the IEEE 21st International Conference on Advanced Information Networking and Applications*, 2007.

[137] Y. L. Sun, W. Yu, Z. Han, K.J., and R. Liu, "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *IEEE Journal on Selected Areas in Communications, 24 (2) : 305-317*, 2006.

[138] A. MacKenzie, "Game theoretic analysis of power control and medium access control," *Thesis*, 2003.

[139] Z. Han, P. C., and K. Liu, "A self-learning repeated game framework for optimizing packet forwarding networks," *Proceedings of the Wireless Communications and Networking Conference*, 2005.

[140] D. B. Johnson, D. A. Maltz, and Y.-C. Hu, "The dynamic source routing protocol for mobile ad hoc networks (dsr)," Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt, 2004.

[141] V. Buskens, *Social Networks and Trust*. Kluwer Academic Publishers, 2002.

[142] D. Fudenberg and J. Tirole, *Game Theory*. The MIT Press, 1991.

[143] K. Siegrist, *Virtual Laboratories in Probability and Statistics*. University of Alabama in Huntsville, 2001. [Online]. Available: http://www.math.uah.edu/stat/

[144] R. Axelrod, *The Evolution of Cooperation*. Perseus Books Group, 2006.

[145] R. Bellman, “On the theory of dynamic programming,” *Proceedings of the National Academy of Sciences*, 1952.