

Digital Forensic Readiness for Wireless Sensor Network Environments

by

Francois Mouton

Submitted in fulfilment of the requirements for the degree

Magister Scientiae (Computer Science)

in the

Faculty of Engineering, Built-Environment and Information Technology

at the

University of Pretoria

January 2012

Digital Forensic Readiness for Wireless Sensor Network Environments

by

Francois Mouton

supervised by

Prof H.S. Venter

Department of Computer Science

Magister Scientiae (Computer Science)

Abstract

The new and upcoming field of wireless sensor networking is unfortunately still lacking in terms of both digital forensics and security. All communications between different nodes (also known as motes) are sent out in a broadcast fashion. These broadcasts make it quite difficult to capture data packets forensically and, at the same time, retain their integrity and authenticity. The study presents several attacks that can be executed successfully on a wireless sensor network, after which the dissertation delves more deeply into the flooding attack as it is one of the most difficult attacks to address in wireless sensor networks. Furthermore, a set of factors is presented to take into account while attempting to achieve digital forensic readiness in wireless sensor networks. The set of factors is subsequently discussed critically and a model is proposed for implementing digital forensic readiness in a wireless sensor network. The proposed model is next transformed into a working prototype that is able to provide digital forensic readiness to a wireless sensor network. The main contribution of this research is the digital forensic readiness prototype that can be used to add a digital forensics layer to any existing wireless sensor network. The prototype ensures the integrity and authenticity of each of the data packets captured from the existing wireless sensor network by using the number of motes in the network that have seen a data packet to determine its integrity and authenticity in the network. The prototype also works on different types of wireless sensor networks that are in the frequency range of the network on which the prototype is implemented, and does not require any modifications to be made to the existing wireless sensor network. Flooding attacks pose a major problem in wireless sensor networks due to the broadcasting of communication between motes in wireless sensor networks. The prototype is able to address this problem by using a solution

proposed in this dissertation to determine a sudden influx of data packets within a wireless sensor network. The prototype is able to detect flooding attacks while they are occurring and can therefore address the flooding attack immediately. Finally, this dissertation critically discusses the advantages of having such a digital forensic readiness system in place in a wireless sensor network environment.

Keywords

broadcasting, communication, digital forensics, digital forensic readiness, flooding attacks, flooding detection, security, sensor networks, wireless networks, wireless sensor networks.

Acknowledgements

I would like to express my sincere thanks to the following people for their assistance during the production of this dissertation:

- My parents, Kobus and Jane Mouton – Thank you for your loving support both throughout my studies and in all the decisions I have made during my life.
- Megan Bester – Thank you for always assisting me with emergency language editing, even at the most inconvenient of times.
- Bertie Viljoen – Thank you for assisting me with last-minute tasks that had to be completed in order to finalise this dissertation.
- Mercia Malan – Thank you for assisting me with last-minute tasks that had to be completed in order to finalise this dissertation.
- Hein Venter – Thank you for your patience, inspiration and motivation. Most of all, thank you for your friendship while I was under your supervision.

Table of Contents

Abstract	i
Keywords	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures	x
List of Tables	xii
List of Equations	xiii
Part I: Introduction	1
Chapter 1 Introduction	2
1.1 Introduction	2
1.2 Motivation	3
1.3 Problem Statement	4
1.4 Goals and Objectives.....	5
1.5 Layout.....	5
Part II: Background	8
Chapter 2 Wireless Sensor Networks	9
2.1 Introduction	9
2.2 Ad hoc Sensor Networks.....	10
2.2.1 Protocol.....	10
2.2.2 Pitfalls and Attacks	11
2.3 Wireless Sensor Networks (WSNs)	12
2.3.1 Graphic representation of a wireless sensor network	12
2.3.1.1 User.....	13
2.3.1.2 Management Server	14
2.3.1.3 Sensor Field	14

2.3.1.4	Wireless Sensor Node (mote)	14
2.3.1.5	Base Station	15
2.3.1.6	Short-range Wireless Communication.....	15
2.3.1.7	Long-range High-speed Communication	16
2.3.2	Wireless Sensor Network Applications	16
2.3.3	Design Factors Specific to a WSN implementation	17
2.3.3.1	Fault Tolerance	18
2.3.3.2	Scalability	18
2.3.3.3	Costs	18
2.3.3.4	Hardware Constraints	18
2.3.3.5	Sensor Network Topology.....	19
2.3.3.6	Environment	19
2.3.3.7	Transmission Media	19
2.3.3.8	Power Consumption	19
2.3.4	WSN Device Communication protocol	20
2.4	Conclusion.....	23
Chapter 3	Attacks on WSNs.....	24
3.1	Introduction	24
3.1.1	Types of Attacks	24
3.1.1.1	Data Interception	24
3.1.1.2	Sinkhole Attacks.....	24
3.1.1.3	Attacks that Require Physical Access to the Motes	25
3.2	General Flooding Attacks.....	25
3.2.1	Ping of Death	26
3.2.2	Smurf.....	27
3.2.3	Echo-Chargen	28
3.2.4	SYN Flood	28

3.3	Wireless Sensor Network-Specific Flooding Attacks.....	30
3.4	Conclusion.....	31
Chapter 4 Digital Forensics and Digital Forensic Readiness		32
4.1	Introduction	32
4.2	Defining Digital Forensics	32
4.3	Digital Forensic Process.....	33
4.4	Digital Forensic Readiness.....	34
4.4.1	Defining Digital Forensic Readiness	34
4.4.2	Achieving Digital Forensic Readiness.....	35
4.4.2.1	Time Period Required to Perform a Digital Forensic Investigation.....	35
4.4.2.2	Cost Involved to Perform a Digital Forensic Investigation.....	35
4.4.2.3	Collecting Evidence without Disrupting the Environment.....	36
4.5	Conclusion.....	37
Part III: Model		38
Chapter 5 Model requirements.....		39
5.1	Introduction	39
5.2	Special WSN requirements	40
5.2.1	Communication Protocol	42
5.2.2	Proof of Authenticity and Integrity.....	43
5.2.3	Time stamping	44
5.2.4	Modification of the network after deployment	46
5.2.5	Protocol Data Packets	47
5.2.6	Radio Frequencies.....	48
5.2.7	Power Supply	48
5.2.8	Network overhead	49
5.2.9	Data Interference.....	49
5.3	Forensic Readiness Requirements for WSNs.....	50

5.4	Conclusion.....	52
Chapter 6	Proposed Model.....	53
6.1	Introduction	53
6.2	Model	53
6.3	Adhering to the digital forensic readiness requirements.....	55
6.3.1	The broadcasted communication from the oWSN should be intercepted in a forensically sound manner.	55
6.3.2	While intercepting communication, there should be no extra network overhead on the oWSN.....	55
6.3.3	The fWSN should by no means be able to influence the oWSN or influence any sensory data transmitted within the oWSN.....	56
6.3.4	The fWSN should not increase the power consumption in the oWSN and the fWSN should have at least the same network lifetime or longer than the oWSN in terms of battery power.	56
6.3.5	The fWSN should be able to capture all possible types of communication that can be sent from the oWSN.....	56
6.3.6	The fWSN should be able to communicate on the same radio frequencies as the ones that the oWSN is capable of using.....	56
6.3.7	All communication within the fWSN should occur on a frequency that is not utilised in the oWSN.....	56
6.3.8	The fWSN should be designed in such a manner that the network topology or the routing protocol used by the oWSN does not influence the fWSN's operation.	57
6.3.9	If an intruder WSN is in the area and communicates on a frequency that influences the oWSN, then the fWSN should be able to forensically capture these data packets.....	57
6.3.10	It should be possible to implement the fWSN without any modification of the oWSN.. ..	57
6.3.11	The authenticity and integrity of all the data packets should remain intact while they are being captured on the fWSN.....	57
6.3.12	The data packets that are captured in the fWSN should be stored in such a way that their authenticity and integrity are not compromised.	57

6.3.13	It should be possible to verify the authenticity and integrity of all the data packets in case a digital investigation takes place.	58
6.3.14	The data packets should have a time stamp assigned to them that does not violate their authenticity and integrity.	58
6.3.15	The sequence of the packets captured should reflect the true sequence in which they were transmitted from the original network.....	58
6.4	Conclusion.....	59
Part IV: Prototype		60
Chapter 7 Prototype Equipment.....		61
7.1	Introduction	61
7.2	Imote2 Sensor Motes.....	61
7.2.1	Imote2 Board	64
7.2.2	ITS400 Sensor Board.....	65
7.3	TelosB Mote.....	65
7.4	.net Micro Framework 2.0.....	67
7.5	Management Server.....	69
7.6	Conclusion.....	69
Chapter 8 Prototype Overview		71
8.1	Introduction	71
8.2	Demonstration Environment	71
8.3	Prototype development.....	75
8.3.1	fWSN Field Motes	76
8.3.2	fWSN Base Station	77
8.3.3	fWSN Management Server	78
8.3.3.1	Packet Logging Software.....	79
8.3.3.2	Packet Analysis Software	80
8.4	Conclusion.....	82

Chapter 9	Prototype Setup	83
9.1	Introduction	83
9.2	Demonstration I: General Conditions	83
9.3	Demonstration II: Flooding Attacks on WSNs	91
9.4	Demonstration III: Sensory Data Verification	98
9.5	Demonstration IV: Capturing of TelosB data packets	102
9.6	Conclusion	106
Part V:	Conclusions	107
Chapter 10	Discussions on Digital Forensic Readiness and Flooding Detection	108
10.1	Introduction	108
10.2	WSN digital forensic readiness requirements	108
10.3	WSN Digital forensic readiness model	109
10.4	WSN digital forensic readiness prototype	109
10.4.1	Digital forensic readiness	109
10.4.2	Flooding detection	111
10.5	Conclusion	112
Chapter 11	Conclusion	113
Bibliography		116
Appendix A:	Source Code	125
A.1	fWSN Field Mote	125
A.2	fWSN Base Station	127
A.3	Forensic Packet Logging Software	129
A.4	Forensic Packet Analysis Software	132
A.5	oWSN Field Mote	138
A.6	oWSN Base Station	142
Appendix B:	Published Papers	145

List of Figures

Figure 1.1. A graphic representation of dissertation layout.....	7
Figure 2.1. A graphical representation of a wireless sensor network (Mouton & Venter, 2009).....	13
Figure 2.2. The sensor networks protocol stack (Akyildiz et al., 2002).....	21
Figure 2.3. Imote2 Wireless Sensor Network Data Packet.....	22
Figure 3.1. Three-Way Connection Handshake.....	29
Figure 5.1. A graphical representation of an oWSN with an overlaying forensic WSN.....	41
Figure 6.1. A graphical representation of an oWSN with an overlaying forensic WSN.....	54
Figure 7.1. Imote2 board.....	62
Figure 7.2. ITS400 sensor board.....	62
Figure 7.3. Imote2 battery board.....	63
Figure 7.4. Imote2 – complete mote with all three boards plugged into one another	63
Figure 7.5. Imote2 – complete mote with perspective of height	64
Figure 7.6. TPR2420 TelosB mote	67
Figure 8.1. Wireless Sensor Network Layout 1	72
Figure 8.2. Wireless Sensor Network Layout 2.....	73
Figure 8.3. Network Layout 1 represented graphically	74
Figure 8.4. A closer look at how fMoteID-201 and oMoteID-101 are spaced apart.....	75
Figure 8.5. fWSN Field Mote Data Packet.....	77
Figure 8.6. Sample log file on 12/07/2009 at 12:38:00 until 12:38:06.....	80
Figure 8.7. Sample GUI for a log file generated on 12/07/2009 at 13:02:07 until 13:06:38.....	81
Figure 9.1. Demonstration I: Wireless Sensor Network Layout for General Conditions.....	84
Figure 9.2. Demonstration I: Wireless Sensor Network Layout for General Conditions (Graphic Representation)	85
Figure 9.3. Demonstration I: Packet analysis software screenshot to demonstrate how the prototype performs under general WSN conditions.....	86

Figure 9.4. Demonstration I: Packet analysis software screenshot with data sorted by the “Originating MoteID” column.....	87
Figure 9.5. Demonstration II: Wireless Sensor Network Layout for Flooding Attack.....	91
Figure 9.6. Demonstration II: The start of the flooding attack.	92
Figure 9.7. Demonstration II: First flooding alert message	94
Figure 9.8. Demonstration II: Impact of flooding on WSNs	95
Figure 9.9. Demonstration II: Continuous flooding alerts.....	96
Figure 9.10. Demonstration II: End of the flooding due to Tk being below the threshold	97
Figure 9.11. Demonstration III: Wireless Sensor Network Layout for Sensory Data Verification...	98
Figure 9.12. Demonstration III: Packet analysis tool screen capture to show the change in light and temperature throughout the oWSN	99
Figure 9.13. Demonstration III: oMoteID-101 in a direct sunlight environment	100
Figure 9.14. Demonstration III: oMoteID-101 placed in a refrigerator.....	100
Figure 9.15. Demonstration IV: Capturing TelosB Data Packets.....	103
Figure 9.16. Demonstration IV: Crossbow TelosB mote with predefined software	104
Figure 9.17. Demonstration IV: TelosB communication captured by the fWSN.....	105

List of Tables

Table 5.1. Factors to take into consideration when implementing digital forensic readiness on an IEEE 802.15.4 wireless sensor network	51
--	----

List of Equations

Equation 9.1. Calculation of the flooding coefficient.....	93
Equation 9.2. Calculation of the number of extra data packets per time interval.....	93
Equation 9.3. Calculation of the flooding in Figure 9.7	94

Part I: Introduction

Chapter 1 Introduction

1.1 Introduction

Technology has become an integral part of our daily lives. We use technologies that work for us in the most extravagant ways – most of which are seemingly integrated with each other through some kind of network and/or the Internet. For example, mobile phones are migrating to the latest generation of mobile phones – the so-called enhanced third generation (3G) – which enables one not only to make voice or video calls, but also to run various kinds of other applications on the device, such as browsing the Web, managing appointments, navigating along streets and through buildings using general positioning system (GPS) software, and much more. Next-generation networks (NGNs), in turn, provide exciting and state-of-the-art possibilities for wireless and ad hoc networking (Orecchia, Panconesi, Petrioli & Vitaletti, 2004; Santi, 2005; Tseng, Ni & Shih, 2003). One particularly exciting NGN includes wireless sensor networks (WSNs), which are discussed in more detail in the dissertation.

Although such innovative technologies are designed to improve our lives, they are unfortunately often the target of malicious intent. Computer viruses and worms probably constitute the most well-known example of such malintent. Although the Internet is traditionally fraught with these menaces, the creators of viruses and worms have also escalated their malintentions to all modern technologies. Viruses and worms, however, can be seen as an explicit and well-known form of malintent that can often be detected very successfully and stopped in its tracks by traditional applications such as antivirus intrusion detection system (IDS) and firewall applications. Due to the nature of these traditional applications, a proactive approach could be adopted of detecting and dealing with malintent such as viruses and worms (Bace, 2000). A proactive approach means that if there is an attempt at malintent, the attempt will be stopped immediately, provided that the application was able to detect it. In a reactive approach, the application will only alert the user of the attempt at malintent and wait for the user to decide if it should be stopped or not.

Besides viruses and worms, however, there are other forms of malintent that cannot easily be dealt with in a proactive manner. For example, enormous amounts of fraudulent credit card transactions are committed over the Internet. Often communications regarding fraudulent transactions are encrypted and therefore cannot be dealt with in a proactive manner. In the late '90s techniques were introduced that would allow law enforcement authorities in possession of a search warrant to retrieve a decryption key from a third party to decrypt communications if such communications were suspected to have malintent – a concept known as key escrowing (Denning &

Smid, 1994). A major flaw of key escrowing, however, is that one's right to privacy is seen to be invaded, which has resulted in key escrowing never really being successfully employed.

As new technologies emerge, such as wireless sensor networks (WSNs), a paradigm shift often takes place with regard to the way in which wireless nodes communicate. To begin with, nodes in WSNs do not function on the same protocol stack as the TCP/IP protocol stack utilised by some of the wireless networking standards such as IEEE 802.11 (Crow, Widjaja, Kim & Sakai, 1997). The main difference is that such nodes do not have fixed media access control (MAC) addresses. This poses a problem in WSNs in that it is very difficult or even impossible to employ proactive detection measures (such as antivirus, IDS and firewall applications) in WSNs. Even though these proactive detection measures exist, they too have difficulty in detecting malintent. For example, due to the fact that WSN nodes do not have fixed MAC addresses (Ye, Heidemann & Estrin, 2002), traditional proactive applications may not be able to detect denial-of-service (DoS) attacks in WSNs. In addition, WSN topology is not fixed and nodes are prone to be mobile, complicating matters considerably for traditional proactive detection measures.

The remainder of Chapter 1 provides the motivation for this research, gives the reader a brief overview of WSNs and discusses the foreseeable problems with WSNs. In the next section the reader is introduced to the problem on which the researcher has focused and is given a brief overview of how the problem will be addressed. Chapter 1 concludes with a layout section that provides an overview of the chapters to follow.

1.2 Motivation

WSNs involve a fairly new technology and as far as the author is aware, no research has been done on WSNs in terms of digital forensic readiness. For the purpose of this study, digital forensic readiness is defined as *the notion to perform a digital forensic investigation in the shortest amount of time with the least amount of cost and without having to disrupt the original network that has to perform mission critical tasks*. In order to explore the notions of digital forensic readiness further in terms of WSNs, the reader is briefly informed of the unique qualities of WSNs.

WSNs are ad hoc networks and, as is the nature of ad hoc networks, their topology can dynamically change throughout the lifetime of the network (Akyildiz, Sankarasubramaniam & Cayirci, 2002). Ad hoc networks are networks in which the devices can communicate with one another without a fixed infrastructure or topology (Wu & Stojmenovic, 2004). The nodes in a wireless sensor network are known as sensors or, as in this dissertation and elsewhere in the literature, referred to as motes. The only way that these motes can communicate with each other is by broadcasting their data in the hope that there is at least one other mote within range to receive

the data packet. The receiving nodes broadcast the received data packet until the packet eventually reaches the gateway (also referred to as the base station), from where the data packets in turn are sent to a management server (Orecchia et al., 2004; Slijepcevic, Potkonjak, Tsiatsis, Zimbeck & Srivastava, 2002). This type of broadcast communication complicates the ability to perform digital forensic analysis on the network.

WSNs have many different applications such as in the military, environmental and health environments. It is of course essential that some kind of forensic readiness system be in place in these environments in case the equipment generates erroneous communication or an attacker attempts to cause malintent within the wireless sensor network. The applications in which WSNs can be used can contain sensitive data and thus it is important to explore the field of digital forensic readiness on WSNs.

Now that the reader has been informed of our motivation for the research and all foreseeable problems with WSNs have been mentioned, the next section explores the problem statement that this study focused on.

1.3 Problem Statement

As the use of wireless sensor network equipment becomes more widespread and used in crucial and sensitive environments (e.g. habit and environment monitoring), there is currently little if any way of applying digital forensics in these environments. In addition, there is a need to implement a digital forensic system alongside existing wireless sensor networks. This poses a further problem. Since WSNs are set up to function in very particular environments, how is digital forensics applied in a WSN without infringing on the setup, functionality and performance of the WSN? In order to solve this problem, this dissertation proposes a digital forensic readiness system that is deployed on a separate set of WSN equipment – literally a second WSN deployed alongside the existing WSN that is to be monitored. For the purpose of this study, the WSN being monitored is referred to as the original wireless sensor network (oWSN) and the WSN executing the monitoring is referred to as the forensic wireless sensor network (fWSN). The fWSN should therefore have the capability to forensically monitor an existing WSN without infringing on it or having any previous knowledge of it. Flooding is another major problem that cannot easily be detected in an existing WSN. This study consequently examined the problem of flooding in WSNs and will provide a possible solution to be incorporated in the forensic system.

The problem as stated can be summarised by addressing the following subproblems:

- There is currently no means of implementing a digital forensic readiness system on wireless sensor networks.
- A digital forensic readiness system should be implemented independent of the oWSN as there is no easy solution for reprogramming and redeploying the oWSN.
- Lastly, the difficult problem of flooding attacks on WSNs also needs to be addressed, as this can severely hamper the performance a digital forensic readiness solution.

The following section provides more detail on the goals that have been set to solve the problems mentioned above and the objectives that this research aims to fulfil.

1.4 Goals and Objectives

The goal of this study is to propose a way to forensically capture the data that is transmitted throughout an existing WSN. The forensic capturing of data implies that it must be captured according to a forensically sound or proven method. An fWSN is introduced for this purpose, and great care is taken to ensure the integrity and authenticity of the data packets captured. However, this dissertation does not focus on the digital investigation process, but rather on the notion of collecting digital forensic evidence by means of a digital forensic readiness system in WSNs. To propose a way to forensically capture the data that is transmitted throughout an existing WSN, we first need to determine the specific requirements of a WSN. For this reason, a model is proposed that sets out the specific requirements to be taken into consideration while trying to achieve digital forensic readiness in a WSN environment.

The system proposed in this study has more uses than just aiding us to conduct a digital forensic investigation. The suggested fWSN has the ability to address other problems in WSNs. In terms of security in WSNs it is currently very difficult to defend a network against flooding attacks. This research shows how the secondary network is able to detect and pinpoint flooding to a specific area of the network.

The following section discusses the layout of the dissertation and provides a brief preview of each of the chapters that follow.

1.5 Layout

Part I is the introduction section and includes this first chapter in which the motivation for the research, the problem statement and goals and objectives of this study are provided.

Part II consists of three background chapters to introduce the reader to all the technologies and concepts that are used in this dissertation. Chapter 2 provides the reader with background on

wireless sensor networks and a brief overview of the communication protocol that is used in WSNs. The background information is intended to familiarise the reader with the broadcasting fashion in which WSNs communicate within the network itself. Chapter 3 focuses briefly on general attacks in WSNs so as to inform the reader of the different type of attacks that can be launched against a WSN, as well as the effects that these attacks can have on WSNs. Next, the reader is introduced to the notion of flooding and shown what a severe impact flooding attacks can have on wireless sensor networks. Chapter 4 provides an overview of digital forensics as it is important to first understand the notion of digital forensics before delving into the field of digital forensic readiness. The focus subsequently shifts to digital forensic readiness, as the main goal of this study is to achieve digital forensic readiness in wireless sensor networks. The reader is also introduced to the benefits that a digital forensic readiness system would have for a WSN.

Part III consists of two chapters. Chapter 5 proposes a model for implementing digital forensic readiness in a WSN environment and discusses the special requirements in a WSN environment as well as how they differ from the requirements of an ordinary IEEE 802.11 wireless connection. The chapter also proposes the model that should be used to implement digital forensic readiness in a WSN environment, since having such a model makes it easier for future implementations of digital forensic readiness systems on WSNs. Chapter 6 critically reviews the model and shows the advantages of having a model in place for digital forensic readiness.

Part IV contains three chapters in which the prototype is proposed and implemented. Chapter 7 introduces the reader to the physical wireless sensor network devices and the equipment on which the prototype is implemented. Chapter 8 discusses the implementation of the prototype on the wireless sensor network devices. In this chapter the hardware and software used to implement the prototype are discussed, while reference is also made to the limitations that were experienced. In Chapter 9 the researcher makes a critically analysis of the prototype by demonstrating certain scenarios and how the prototype performs in these scenarios.

Part V contains the two concluding chapters. Chapter 10 discusses the effectiveness of the prototype and model, and critically discusses all the research that has been completed in preparation for this dissertation. The chapter also discusses the feasibility of using a secondary network to achieve forensic readiness in a WSN environment. Finally, Chapter 11 contains a brief summary of the extent to which the research problems have been solved and addressed in this dissertation. The chapter is concluded by proposing areas of further research.

Chapter 1 is now concluded with a graphic representation (Figure 1.1) of the layout of the dissertation.

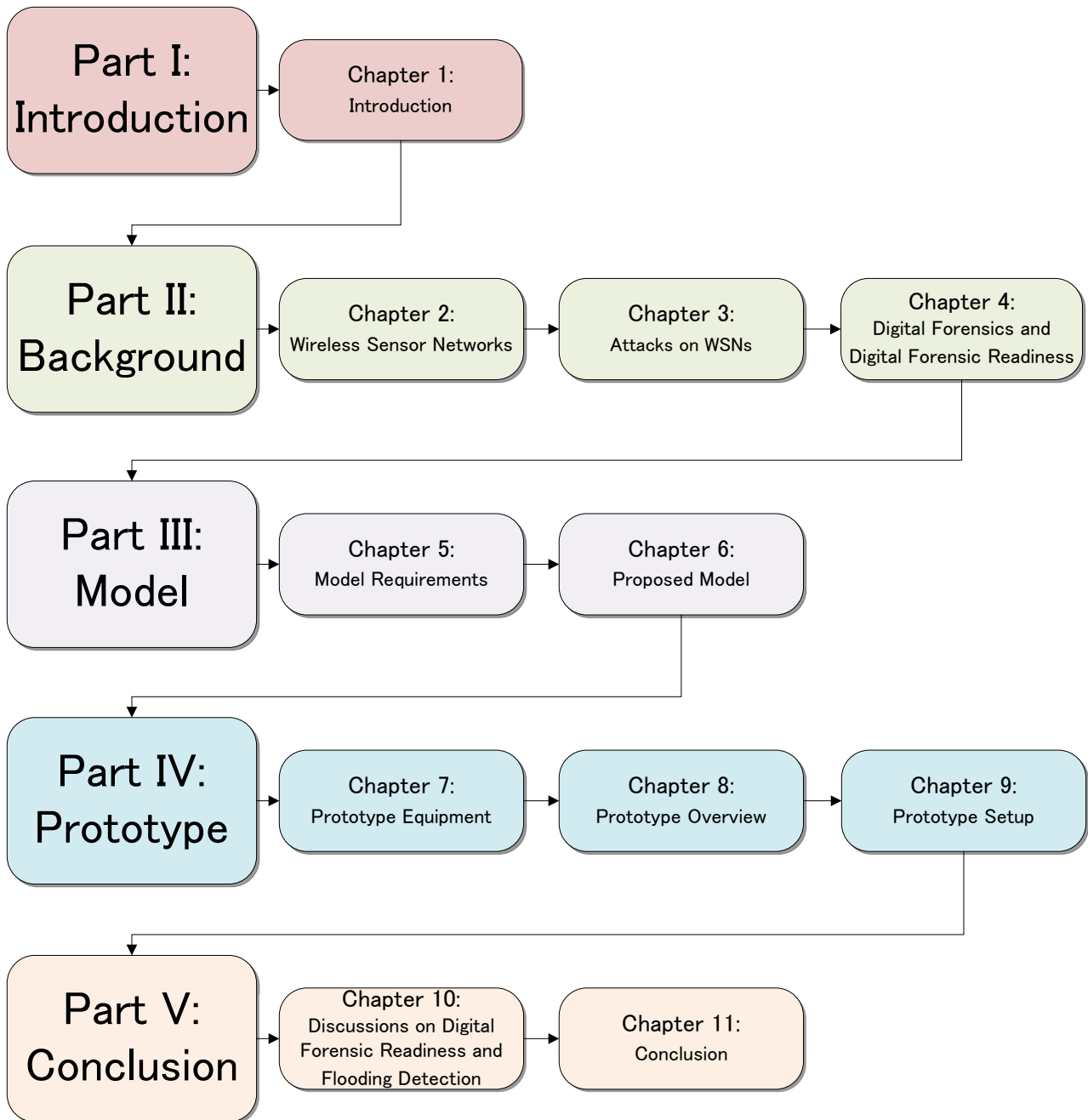


Figure 1.1. A graphic representation of dissertation layout

The next part (Part II) of the dissertation covers the background sections that deal with wireless sensor networks, flooding attacks, digital forensics and digital forensic readiness.

Part II: Background

Chapter 2 **Wireless Sensor Networks**

2.1 Introduction

Wireless Sensor Networks (WSNs) still constitute a relatively new area of research in computer science and the first papers on WSNs were published around the start of the 21st century only (Chong & Kumar, 2003; Mouton & Venter, 2009; Akyildiz et al., 2002; Zhu, Setia & Jajodia, 2006; Chen, Jiang & Liu, 2005). Much of the research on WSNs dealt with new areas of application aimed at supporting our modern lifestyle. There has been very little focus on WSN security – especially digital forensics – as this is still a developing technology.

The initial development of WSN devices focused extremely strongly on processing efficiency, while still being able to communicate in an ad hoc fashion (Ye, Heidemann & Estrin, 2002; Wander, Gura, Eberle, Gupta & Shantz, 2005). Research on securing communication between the devices received little attention as most of the focus had been devoted to saving power seeing that WSN motes run on battery power. It is a well-known fact that any type of encryption algorithm would require more processing and thus effectively cause more power drain on the device itself (Akyildiz et al., 2002; Wander et al., 2005).

Since in this day and age security on electronic devices has become an extremely important factor to consider, one would now need to implement some form of security on these devices as they have been developed mostly without taking security aspects into account. Quite a number of researchers have started exploring different techniques on how to implement secure communication protocols on these devices. Researchers such as Perrig and Mouton focused some of their research on implementing security on WSN devices (Perrig, Stankovic & Wagner, 2004; Perrig, Szewczyk, Wen, Culler & Tygar, 2001; Mouton & Venter, 2009). Most of these proposed security solutions considered a type of communication protocol that incorporates one or more security elements.

If the security on these WSN devices had been disregarded to such an extent, one may almost assume that digital forensics and digital forensic readiness on these devices also had to take a backseat. It is for this very reason that this study focuses mainly on digital forensic readiness, as well as on comments on digital forensics and security on these WSN devices. It would be much better to examine these factors while WSN technology was still in its developing stages, as it would become an even more expensive exercise to add these features later.

This chapter now continues with a brief overview of ad hoc sensor networks, after which the focus is shifted to WSNs specifically. The WSNs section focuses on how the devices operate and

which real-world applications can benefit from using WSNs. The chapter is concluded by a brief overview of WSNs and the main pitfalls involved.

2.2 Ad hoc Sensor Networks

Sensor networks are used quite widely in order to improve our everyday lives. A simple example of such a network in action is in most modernised parking bays (Chinrungrueng, Sunantachaikul & Triamlumlerd, 2007). Each parking bay is equipped with a pressure sensor, which is in turn connected to a light in the roof of the parking bay. This light would in normal operation indicate green if the parking bay is unoccupied, but if a vehicle is parked inside the bay, the light would turn red. The red light is visible from a distance and shows that the bay is occupied.

The following subsections introduce the reader to the protocol of ad hoc sensor networks, after which the pitfalls and attacks of the protocols are discussed.

2.2.1 Protocol

The way that ad hoc sensor networks communicate is quite different to the way an ordinary IEEE 802.3 Ethernet network (LAN) communicates. In an IEEE 802.3 Ethernet, the TCP/IP stack is mostly utilised for communication to occur (Spurgeon, 2000; Walrand, 1998). Also, the IEEE 802.3 Ethernet protocol relies on the usage of media access control (MAC) addresses to uniquely identify the devices on the network. In a case where two devices have the same MAC address, it will cause these two devices to behave irregularly on the network (Khoussainov & Patel, 2000). The same applies to an IEEE 802.11 wireless network – when two devices have the same MAC addresses, these devices would behave irregularly (Guo & Chiueh, 2006).

Ad hoc sensor networks rely on broadcasting for communication to occur (Ni, Tseng & Sheu, 2001). This allows the devices to dynamically determine their network layout and network topology (Akyildiz et al., 2002). Then, by means of the routing protocol used, the devices determine how to pass on communication towards a management server (Akyildiz et al., 2002). Depending on the routing protocol used within the sensor network, the devices will know or not know the path to the base station. In the case where the path to the base station is known, only the devices that are known to be closer to the base station would retransmit the data packet in a broadcast fashion. In the case where the path is not known, all the devices would retransmit the data packet in a broadcast fashion only once. However, this makes these devices prone to flooding attacks, due to the fact that they are designed to propagate messages throughout the network in order for the communication to reach the management server.

Broadcasting is an elementary operation that is used by mobile ad hoc networks (Tseng, Ni & Shih, 2003). It occurs when the nodes are transmitting data wirelessly. The individual devices have no previous knowledge of the relationship with their neighbouring devices. Once the information has been broadcast, all the devices in the vicinity will receive this transmission. Upon receipt, the neighbouring devices will analyse the data packet. After the data packet has been analysed, the device will decide (based on the routing protocol used and the destination of the packet) whether to retransmit or drop the data packet.

Any of the devices can emit a broadcast at any given moment. It is not guaranteed that the information sent will reach any of the devices in its designated perimeter. Broadcasting is also unreliable because if another device sent out a broadcast at the exact same moment, messages would clash and both communications would be dropped. The very same message consequently has to be sent multiple times. According to Tseng (2003), the MAC specification does not allow for the acknowledgement of broadcasted information, as it would lead to severe flooding of the network if an acknowledgement packet were to be sent for every data packet received.

Having shown that the communication within an ad hoc sensor network is transmitted in a broadcast fashion, it is important to examine the pitfalls of broadcasting messages, as well as the attacks that this type of protocol is prone to.

2.2.2 Pitfalls and Attacks

According to Tseng, Ni and Shih (2003) and Orecchia et al. (2004), broadcasting has many flaws as it could cause flooding, redundancy or collisions on the network. Research was therefore conducted on secure routing (Karlof & Wagner, 2003) and on using ad hoc subnetworks (Sohrabi, Gao, Ailawadhi & Pottie, 2000) to overcome the broadcasting problem. Still, some of these solutions can hamper the battery life of the sensor devices. This would be due to the increased overhead processing, as secure routing requires extra processing that in turn leads to increased battery drain.

Another pitfall of broadcasting is that all information sent can be intercepted by rogue devices in the network. It would be quite easy for an attacker to intercept broadcasted information and use it against the existing sensor network to perform a whole range of attacks. These attacks range from flooding attacks, data interception attacks or even sinkhole attacks (Zhu, Setia & Jajodia, 2006). Flooding attacks occur when a rogue node intercepts a legitimate data packet and attempts to resend this packet continuously and repeatedly into a sensor network, or when a rogue node injects randomised data packets continuously and repeatedly into a sensor network. A sinkhole attack occurs when a rogue device masquerades itself as a base station or a device that is closest to the base station. This rogue device would then cause the sensor network to transmit data packets to the

rogue device itself as the sensor network would be under the false impression that the rogue device is part of the ad hoc network (Zhu, Setia & Jajodia, 2006). This rogue device could subsequently drop all the data packets in order to perform a denial-of-service attack on the sensor network.

This section provided the reader with a brief introduction to ad hoc sensor networks – wireless or wired. The next section focuses specifically on wireless sensor networks.

2.3 Wireless Sensor Networks (WSNs)

WSNs belong to the general family of ad hoc sensor networks that use multiple distributed sensors to retrieve data from various environments of interest. Chong and Kumar (2003) provide a history on previous accomplishments of WSNs and show how they have evolved in terms of sensing, communication and computing. WSNs consist of wireless nodes with embedded processors and are a form of ad hoc network (Estrin, Girod, Pottie & Srivastava, 2001) with the distinct property that it involves wireless communication (Ye, Heidemann & Estrin, 2002). Mouton and Venter (2009) define a WSN as an ad hoc network that consists of tiny wireless and resilient computing nodes, also known as motes or sensors. These motes are extremely efficient with regard to power consumption and can communicate effectively with other motes within their vicinity (Mouton & Venter, 2009).

In the following subsection, a graphic representation is provided so that the reader has a good overview of WSNs, after which more focus is shed on the detail of how a wireless sensor network operates. The environments in which WSNs can be used are briefly discussed in a second subsection, followed (in the third subsection) by the factors that need to be taken into consideration while designing WSNs. A fourth subsection discusses the WSN communication protocol which is a very important factor in WSNs and for the remainder of this dissertation.

2.3.1 Graphic representation of a wireless sensor network

It is important to give the reader an overview of what a typical wireless sensor network looks like in order to form a good idea of how the devices would operate. A graphical representation of a wireless sensor network is therefore provided in Figure 2.1 (Mouton & Venter, 2009; Heinzelman, Kulik & Balakrishnan, 1999; Sohrabi et al., 2000).

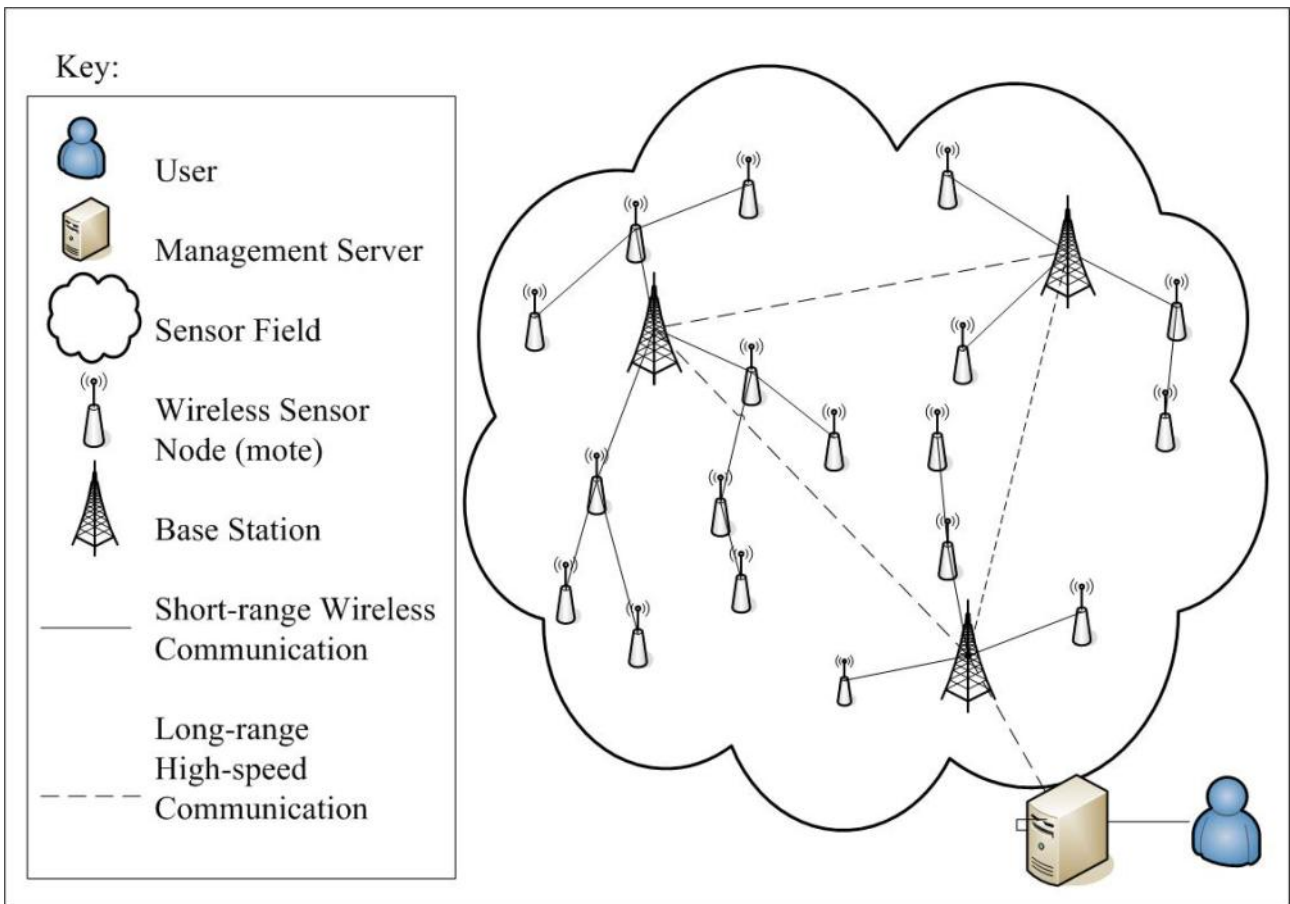


Figure 2.1. A graphical representation of a wireless sensor network (Mouton & Venter, 2009)

Each component in Figure 2.1 plays an integral role in establishing a complete wireless sensor network. The following subsections are devoted to explaining each of the components depicted in Figure 2.1 in detail.

2.3.1.1 User

The user can interact with the WSN through the management server in two possible ways: 1) directly, i.e. physically sitting in front of the management workstation, or 2) remotely using wireless sensor network specific software. This allows the user to provide input or receive information from the WSN through the management server. It also enables the user to dynamically monitor the measurements taken by the motes.

The role of the user is normally taken by a system administrator who has ownership of the WSN. It is the system administrator's task to ensure that any problems arising within the WSN are addressed promptly and resolved. Such problems could range from battery drainage on devices to physical destruction of devices, or even unauthorised removal of devices.

2.3.1.2 Management Server

The management server serves as an interface to the WSN. It is the task of this server to capture and log the information that is received from the base stations. The management server could possibly be connected to multiple base stations if the need arises. This would occur if there are different independent sensor networks in the area and the management server were connected to base stations from both WSNs. One such scenario where a management server can be connected to two different base stations is where the one sensor network is used for forest fire detection and the other for tracking movement inside the forest. One would then have one sensor network aimed solely at detecting forest fires, whereas the other sensor network would detect vibrations in order to track movement.

In our graphic demonstration the management server is connected to a single base station only. However, this single base station is connected to the other base stations in the WSN by means of long-range high-speed link. This link between the base stations can be a physical wire or wireless devices that are configured in a point-to-point fashion. The point-to-point wireless communication does not interfere with the sensor network, as the communication occurs on a different frequency and is directed communication. Directed communication occurs when two wireless antennas are pointed towards one another and in essence form a beam for communication to occur.

2.3.1.3 Sensor Field

The sensor field denotes the physical boundaries of the WSN. Each device within the WSN has its own sensor field as well, indicating the physical range of how far the broadcasted message could possibly travel. This range would depend on the signal strength of the data packet transmission.

2.3.1.4 Wireless Sensor Node (mote)

The wireless sensor nodes in the graphic representation are the devices that affect the actual monitoring. In most WSN research they are referred to as motes. Each of these motes are equipped with a set of sensors that are able to monitor various ambient conditions, physical displacement or event-related conditions surrounding them, and report this information back to the base station. Each mote may be equipped with a different set of sensors and thus each mote could potentially monitor different attributes. The sensors are placed onto a sensor board connected to the WSN device; thus a specific WSN device could use different sensor boards based on the requirement of the WSN.

A mote can also receive information from its surrounding motes, which it would pass on to the base station. This would occur when an individual mote is outside the receiving range of the

base station for communication to occur directly and needs another mote within communication range to convey the data packet to the base station. In this scenario, the mote that is closer to the base station would act as a repeater. A mote would know if it is closer to the base station if the communication protocol that it is using caters for it (Dunkels, Osterlind & Zhitao, 2007). Otherwise, a mote would just rebroadcast the received communication in the hope that it reaches the base station (Dunkels, Osterlind & Zhitao, 2007).

A mote is powered by a battery pack and thus has limited processing power. The communication range of a mote is fairly small, as a stronger radio signal would require more battery power and subsequently decrease the time the mote can stay active without replacing the battery pack.

2.3.1.5 Base Station

A base station serves as a gateway node through which the information of the motes has to travel to reach the management server. The base station has much more processing power than the wireless motes as they are required to receive data packets from all their surrounding motes. Base stations typically use an external power source, as they potentially constitute a single point of failure for a WSN. All communication should eventually travel through the base station towards the management server, and should the base station fail (due to depleted batteries, if the base station runs on battery power), all communications of the motes surrounding that base station will be lost.

A base station may be connected to other base stations by a long-range high-speed link if there is more than one base station within the WSN, since they are sometimes situated far from each other. If there is only one base station in the WSN, it will communicate directly with the management server. Communication between base stations can occur on either a wired or wireless link, based on the distance between the base stations.

2.3.1.6 Short-range Wireless Communication

Short-range wireless communication links are established between neighbouring motes and the neighbouring base stations. This type of wireless communication occurs between a single mote and all its neighbouring motes and base stations within the transmission range of the transmitting mote.

Base stations would always be connected with long-range high-speed links as they would be too far apart to be connected with short-range wireless communication. This is because each segment of the WSN normally has its own base station and the motes communicating to the one base station would not be in range of another base station.

2.3.1.7 Long-range High-speed Communication

Long-range high-speed communication links are established between the separate base stations and the management server for the WSN. These links commonly occur over a physical wire. In cases where this is not possible, the base station has wireless links for reaching the other base stations. There will be no interference with the sensor field nodes as the long-range high-speed communication from the base stations can be established using point-to-point directed antennas that operate on a different frequency. Power consumption should not be an issue with the long-range high-speed communication since the base stations normally have access to an external power source and should not rely on battery power.

This concludes the discussion of the components of a wireless sensor network. As we have now discussed how a typical WSN is deployed and how it operates, it is important to consider the uses of WSNs. The next section suggests scenarios in which it would be applicable to use WSN devices to perform certain tasks so as to provide the reader with some real-world scenarios.

2.3.2 Wireless Sensor Network Applications

WSNs can be used in many environments. Their nodes may consist of many different types of sensors, such as thermal, visual, infrared, radar or acoustic, to name just a few. These nodes can monitor a wide variety of ambient conditions, such as humidity, pressure, sound, noise levels, temperature, lighting conditions and objects moving through a designated area (Elson & Estrin, 2001; Kahn, Katz & Pister, 1999).

Some applications of WSNs include military applications such as the tracking of moving objects and battlefield surveillance (Zhao, Shin & Reich, 2002). In tracking applications, the sensors rely on vibrations caused by movement. The WSN sensors are able to sense the slightest of vibrations. These sensory bits of information are then passed on to the base station which in turn passes them on to the management server. By examining the origin of the vibrations, the management server is able to pinpoint, by means of triangulation, the exact position of the object that is moving and its actual movement. Triangulation can be calculated if the distance is known between two devices that are communicating with a third device (Hartley & Sturm, 1997; Xiang-Yang, Calinescu, Peng-Jun & Yu, 2003; Chun-Hsien, Kuo-Chuan & Yeh-Ching, 2007). One would then determine the angle of communication from each of the two devices towards the third device. Using both the distance between the devices and the angles from both devices towards the third device, one could use trigonometry to calculate the position of the third device. Thus, once the exact locations of all three devices are known, one can determine the position of the object and the direction of its movement by examining the measurements taken by the devices.

Environmental applications include habitat monitoring, forest fire detection and flood detection (Mainwaring, Culler, Polastre, Szewczyk & Anderson, 2002). Consider for instance the detection of forest fires. To perform forest fire detection, the motes are scattered in the forest so that they are within range, forming the sensor field. Within this sensor field, the temperature sensor would be closely monitored. This would allow one to immediately detect the outbreak of a forest fire, to determine by means of triangulation where the forest fire originated and, thus, to determine the fire's location. Afterwards, one would obviously be required to replace the motes that have been destroyed in the forest fire.

WSN applications in health care include the tracking and monitoring of doctors and patients in hospitals, as well as drug administration in hospitals (Lu, Blum, Abdelzaher, Stankovic & He, 2002). For instance, if the WSN devices are embedded into the clothing of patients, one would be able to track the movement, location, heart rate and temperature of the individuals who are wearing the clothes with these devices. Tracking could be implemented in the form of a wrist band if the clothes of doctors/patients need to be changed on a continuous basis.

WSNs can also very easily be used for home and building automation applications (Wheeler, 2007; Callaway, Gorday, Hester, Gutierrez, Naeve, Heile & Bahl, 2002). As the signals to operate circuitry are very elementary requests, these requests could easily be transferred over WSN devices. The small size of WSN devices also helps to keep these devices discreet in home and building automation applications.

Several other applications for WSN devices are also possible. Only the most prominent ones have been discussed above. It is important to note that each of these applications has certain constraints and for each, the significance of some of the design factors within WSNs would differ. For example, the type of sensors and battery packs would differ for different WSN applications. Also, depending on the WSN application, the motes would be either densely scattered or only a few would be placed in specific locations. The next section introduces the reader to these factors, after which each is individually considered.

2.3.3 Design Factors Specific to a WSN implementation

While designing a WSN, one has to take into account many definitive factors as these devices have several constraints. The factors are important because they serve as a guideline to design an application or a protocol for wireless sensor networks.

The factors to be taken into account are fault tolerance, scalability, production costs, hardware constraints, sensor network topology, environment, transmission media and, most importantly,

power consumption (Savarese, Rabaey & Beutel, 2001; Akyildiz et al., 2002). The following subsections discuss each of these factors in detail.

2.3.3.1 Fault Tolerance

Some of the motes may fail or be blocked when they are scattered in the area for which they are designated. They could even be physically damaged or subject to environmental interference. It may even happen that someone physically destroys or removes a mote from the WSN. When a mote fails due to power constraints, there should be other motes that can take over the failed mote's task.

The failure of some of the motes should nevertheless not affect the overall purpose of the sensor network.

2.3.3.2 Scalability

The sensor network should not be affected, depending on the number of motes in the area. In some applications like object tracking only a few motes would be required as one would only need to cover a specific track. This is in contrast to forest fire detection where the area to cover is much larger and would require more motes to cover the entire region of the forest. The same efficiency of the communication protocol should be seen in all the different applications of the network, whether it is a small WSN or a large WSN.

2.3.3.3 Costs

The cost of the entire WSN should not exceed the cost of any other type of solution. For example, it should be cheaper to implement a WSN to detect forest fires than to physically hire people to stand guard and watch for forest fires over a set period of time. In summary, the cost of the WSN should be feasible.

2.3.3.4 Hardware Constraints

The size of the actual mote versus its battery lifetime is an issue to consider while designing WSNs. The mote must be small enough to work in the specific environment, but it must also have sufficient battery life to work throughout the duration of the network's existence.

It is important to note that the biggest physical part of a WSN mote is the battery pack, thus decreasing the size of the entire device would also mean a smaller battery pack, which would provide less power.

2.3.3.5 Sensor Network Topology

It should be considered how the motes would link up to each other right after they have been deployed. A wireless sensor network works on an ad hoc principle (Lewis, 2005). The ad hoc principle allows the motes to dynamically link up and form their own topology when powered on in the wireless sensor network (Maroti, Kusy, Simon & Ledeczi, 2004). An ad hoc network also has the functionality of a self-healing mesh (Maroti et al., 2004). This allows the topology of the network to change dynamically when some of the motes fail, run out of power or if new motes are introduced to the network.

Different sensor network topologies would need to be used in different environments. If the motes are placed in strategic places where they would remain stationary throughout the lifetime of the network, the network topology would only need to be configured once. However, in the case where the motes continuously move around, the network topology will be reconfigured on a continuous basis due to the ad hoc nature of wireless sensor networks.

2.3.3.6 Environment

It should be considered what environmental conditions will have an effect on the sensor network. The motes can be in a stationary position, for example when they are scattered in a forest to detect forest fires. In this case the devices will use less battery power due to the network topology staying constant and the motes not moving around. Motes can also be attached to moving vehicles or be placed in a river stream. When the motes are placed in a stream, they will float downriver past several base stations. Hence, the network topology will have to change dynamically as the motes will have to transmit to several different base stations throughout their lifespan. This would make the environment change dynamically.

2.3.3.7 Transmission Media

It should be determined on what radio frequency the motes would communicate with each other. Transmission should be tested beforehand to ensure that there is no interference from another radio frequency in the designated area.

2.3.3.8 Power Consumption

The mote can only be equipped with a limited power source (Akyildiz et al., 2002). This requires that the power consumption of the communication between the motes and data processing on the mote itself be kept to a minimum (Ye, Heidemann & Estrin, 2002).

One particular issue with WSNs is that all the communication between the nodes is broadcast. As we have seen earlier, this allows other rogue nodes in the vicinity to retrieve such communication and use it for malicious intent. Due to this problem, the entire section that follows has been devoted to providing background information on the WSN communication protocol.

2.3.4 WSN Device Communication protocol

The communication protocol in a WSN is very unique. It relies fully on broadcasting to transmit data from one mote to another. Although several research papers have been published on improving the communication protocol of WSNs (Zhu, Setia & Jajodia, 2006; Orecchia et al., 2004; Tseng, Ni & Shih, 2003), securing the transmissions only received attention in later research (Mouton & Venter, 2009; Perrig et al., 2001). All of the earlier research papers on WSNs (before the security aspects were considered) are mainly concerned with routing paths from a single mote to the base station and back.

The different routing protocols are still susceptible to interception and, thus, information can be falsified and propagated throughout the WSN. This is done when a legitimate packet is intercepted and continuously retransmitted inside the WSN by a rogue mote or device. In fact, the only advancements made in the field of WSNs involve ways to propagate data packets in the quickest, most efficient way towards the base station with minimal disruption to the network, in the case that a mote fails and the route has to be altered.

It is important to note that on most routing protocols the communication between the devices is unencrypted (Akyildiz et al., 2002). Encryption is quite a processing-intensive task and will thus severely hamper the lifespan of the mote, due to the increased processing requirements that will drain more battery power (Perrig et al., 2001; Wander et al., 2005). Also, the encryption of data packets on the fly will take too long and be a tedious task due to the limited processing power of the motes (Wander et al., 2005). Most of the routing protocols consequently use unencrypted transmissions between motes. However, even with the use of secure routing protocols, as proposed by Tseng, Ni and Shih (2003) and Mouton and Venter (2009), the communication between devices can still be intercepted. Using secure routing protocols only makes it more difficult to interpret the packets. Because WSNs communicate in a broadcast fashion, it is impossible to transmit data packets without exposing them to potential interception. Thus, this study assumes that the routing protocols have no real impact on the way that data can be intercepted, as it is believed that all transmission could be intercepted due to the broadcasting nature of WSNs.

Following the general overview of the communication protocols within WSNs, it would be of use to briefly examine the protocol stack for WSNs as proposed by Akyildiz et al. (2002). The sensor networks protocol stack is depicted in Figure 2.2.

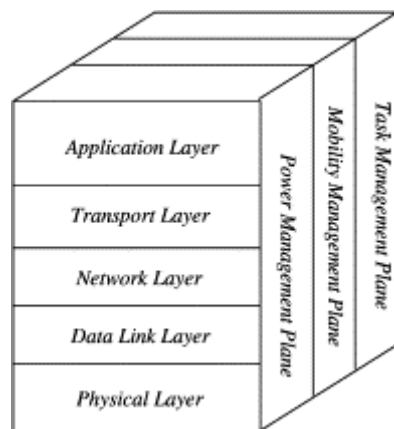


Figure 2.2. The sensor networks protocol stack (Akyildiz et al., 2002)

Figure 2.2 shows that the protocol stack of a sensor network exists of five layers, namely the application layer, transport layer, network layer, data link layer and physical layer. Each of these layers also exists on three planes, namely the power management plane, the mobility management plane and the task management plane.

The application layer is where the application software for the specific sensing task is developed and used. The application layer is basically the interface of the wireless sensor device. The transport layer is used to control the flow of data whenever the application layer requests it. The network layer is used to control the routing protocol when it receives data from the transport layer. The data link layer is responsible for the reliability of the communication and thus handles all access control and error control of the communication. The physical layer is host to the actual transmission devices that are used to transmit the data.

In addition to all of these layers, the power, mobility and task management planes are used to monitor the power, movement and task distribution among motes. The ultimate goal of these planes is to assist with the sensing tasks of the motes and to ensure that the overall power consumption is kept to a minimum.

An in-depth discussion of each layer and plane has been excluded from this background section as the current study does not delve into a level where the protocol stack needs to be altered. For the purposes of this study we used the application layer that is already loaded on the devices to transmit data packets.

The data packets as used in our research are illustrated in Figure 2.3.

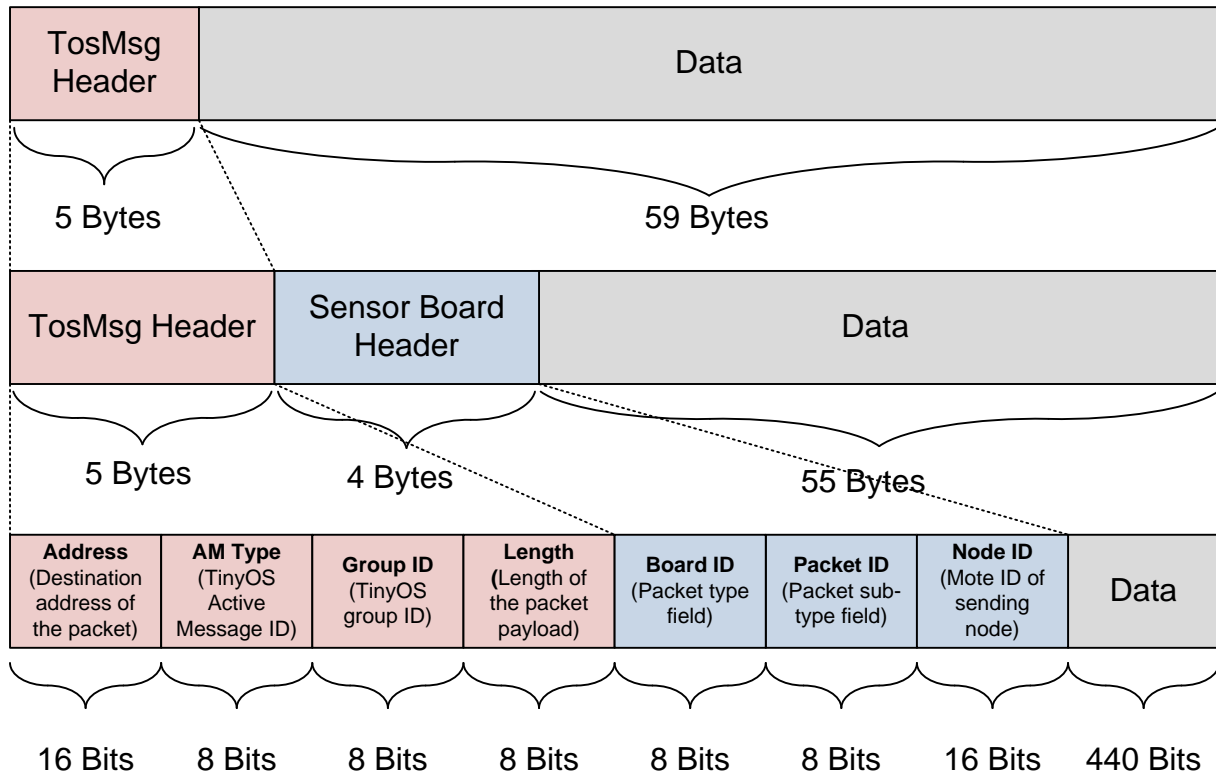


Figure 2.3. Imote2 Wireless Sensor Network Data Packet

The Imote2 WSN data packet that is illustrated in Figure 2.3 is used as an example, as Imote2 WSN devices will be used in the rest of the study. These devices are discussed in more depth in Chapter 7, where the reader is introduced to the devices that were used for the purpose of this dissertation. The first 5 bytes can be seen as the header of the packet, and the remaining 59 bytes can be used to carry data. Furthermore, the TosMsg header can be broken down into four different fields, namely the address block, the active message type block, the group ID block and the length block. The address block contains the destination address for which the data packet is intended. Both the active message block and the group ID block contain unique identifiers for the data packet, which are generated by the TOSRadio protocol (the communication protocol that the Imote2 WSN devices use to communicate). The length block, which indicates the number of bits in the entire data packet, is also generated by the TOSRadio protocol.

The data field, for purposes of this research, is divided into two blocks. The first block contains data about the sensor board that is transmitting the data packet, and the other 55 byte block is completely devoted to data bits. The sensor board header block is subdivided into a board ID block, a packet ID block and a node ID block. The board ID and the packet ID blocks contain only information about the packet being transmitted or sometimes they are combined to be used as a 2-

byte sequence number, whereas the node ID block contains information about the unique identifier of the mote transmitting the data.

The next section concludes this chapter by highlighting certain aspects referred to further on in this dissertation.

2.4 Conclusion

Chapter 2 introduced the reader to a graphical representation of WSNs and gave an in-depth explanation of the functionality of each of the components within a WSN. The different real-life applications of WSNs were also discussed and have shown the benefit that these devices can have for our everyday lifestyle.

Some design constraints were also examined and consideration was given as to why they are important in designing a WSN implementation. Lastly, focus was shifted to communication between WSN devices as this plays a pivotal role in this study.

Some specific problems with WSNs still need to be addressed, one of which involves flooding attacks on WSNs. Flooding attacks are one of the hardest attacks to thwart on WSNs. This is mainly due to the broadcasting nature of WSNs. The next chapter is therefore designed to introduce the reader to flooding attacks, show how they can severely hamper WSNs and suggest a means of minimising the effect of flooding attacks on WSNs.

Chapter 3 **Attacks on WSNs**

3.1 Introduction

WSNs are a fairly new technology with many real-life applications. Even so, it is prone to a number of known attacks that can render the entire network ineffectual (Perrig et al., 2001). Flooding attacks, for instance, can severely hamper WSNs and can influence the way one would approach the task of implementing digital forensic readiness.

This chapter introduces the reader to three typical types of attacks on WSNs, after which more light is shed on a fourth much more important type of attack, i.e. a flooding attack. A flooding attack is one of the most difficult attacks to counter and subsequently constitutes the main attack that this study focuses on.

3.1.1 Types of Attacks

Many different types of attacks can be launched at wireless sensor networks. Our focus will be on three of these attacks, namely data interception, sinkhole attacks and attacks that require physical access to the nodes.

3.1.1.1 Data Interception

Each node broadcasts the data it has gathered from its neighbouring nodes in an attempt to get the data to reach the other nodes. Currently, as far as the researcher is aware, there is no encryption on the data packets which are broadcast throughout the WSN (Akyildiz et al., 2002). This allows a hostile entity to plant its own WSN or other hostile interception equipment in the sensor field. The hostile WSN can then eavesdrop on any communication and will pass on any intercepted data to the hostile management server (Slijepcevic & Potkonjak, 2001). This could disclose valuable information belonging to the legitimate WSN.

3.1.1.2 Sinkhole Attacks

This is one of the most difficult attacks to deal with within wireless sensor networks (Zhu, Setia & Jajodia, 2006). In a sinkhole attack, an adverse node will try to attract information from its neighbours. It would then either pass this information on to a hostile entity or simply drop the packets. This would lead to a DoS (denial-of-service) attack on the legitimate WSN. Wood and Stankovic (2002) define a DoS attack as any event that diminishes or eliminates a network's capacity to perform its expected function.

Sinkhole attacks are only effective if the WSN uses a name-based communication protocol in which the nodes are aware of the fastest path to the closest base station (Dunkels, Osterlind &

Zhitao, 2007). This attack is accomplished by broadcasting false information about the position of the next closest base station to the neighbouring nodes. All the nodes in that vicinity will simply assume that this information is correct and would not attempt to send data directly to the legitimate base station. This would allow the hostile node to intercept all the network packets.

3.1.1.3 Attacks that Require Physical Access to the Nodes

Most research papers that we have dealt with have very little emphasis on these types of attacks. It is very important to note that an intruder could physically connect his workstation to a node and transfer all the code that is on the node directly to his workstation. When physically connecting to a node in order to access the code on the node, no authentication mechanisms are in place.

The attacker could even take the code it received, modify it and deploy it back onto the node. This is a very powerful attack as it may look as if a legitimate node is sending legitimate communication, whereas the communication is in fact falsified. The attacker could go as far as using a covert channel to send the real sensory data to one of his nodes so as to steal this correct information and possibly use it for malicious intent. Such an attack could be very harmful to one's network and is very difficult to identify.

So far, some general types of attacks on WSNs have been discussed. The goal of the remainder of this chapter is however to introduce the reader to the more important type of attack – a flooding attack – and then to focus specifically on the impact of flooding attacks on WSNs. Section 3.2 focuses on general flooding attacks, while Section 3.3 focuses on WSN-specific flooding attacks. An overview of flooding attacks will afterwards conclude the chapter.

3.2 General Flooding Attacks

Consider the following scenario that has had many an organisation in disarray in the past. Suddenly the web server, which is one of the mission critical applications in the corporate environment, is not available anymore. For every minute that it is down, the organisation may start losing huge amounts of money.

In layman's terms, this just means that the organisation has become one of the victims of a DoS (denial-of-service) attack (Wood & Stankovic, 2002; Mirkovic et al., 2004). A flooding attack is just one form of a DoS attack (Cheswick, Bellovin & Rubin, 2003; Pfleeger & Pfleeger, 2006; Perrig, Stankovic & Wagner, 2004). The typical flooding attack is quite an elementary DoS attack, as it simply bombards the network with excessive data packets, which confuses the target network and often causes the network to crash (Moore, Shannon, Brown, Voelker & Savage, 2006; Karig & Lee, 2001). The more sophisticated flooding attacks make use of the Internet Control Message

Protocol (ICMP), which is normally used for system diagnostics (Pfleeger & Pfleeger, 2006). All of these ICMP protocols have an important use as far as network management is concerned, but they can also be used for malicious intent when a DoS attack is planned. The ICMP protocols include the following (Pfleeger & Pfleeger, 2006):

- *Ping*, which requests a reply response from another device on a network to determine if the other device is alive on the network;
- *Destination unreachable*, which would indicate that a ping request failed and thus a destination unreachable response is returned;
- *Echo*, which is very similar to a ping request. The Echo request sends data to another machine on the network, to which the latter should reply with the same data payload. This allows one to establish if the other machine is alive and how reliable the network connection between the two machines is;
- *Source quench*, which would mean that the machine to which data is being sent, is receiving too much data and that the source machine should wait a while before sending any data packets again.

The subsections that follow are dedicated to examining how these ICMP protocols can be used to perform a DoS attack.

3.2.1 Ping of Death

The default ping program, as mentioned earlier, is used to determine if a machine on the other end of the network is able to respond, in a bid to confirm successful network connectivity. The ping-of-death attack can thus utilise the echo protocol to send data packets along with the ping request. With the combined use of both ICMP and ping, one is able to determine the network distance to the host in milliseconds (Harris & Hunt, 1999). This shows that this type of ping request could be used for legitimate reasons.

IP datagrams cannot exceed a maximum size of 65 535 bytes (Harris & Hunt, 1999). The term 'ping of death' is used for any ping packet with a packet data of size that is large enough for the receiving machine to cause it to halt or go into a long reset cycle (Templeton & Levitt, 2000). The attack causes such systems to halt when the sent data packets are larger than what the network connection of the target machine can handle, i.e. when the size of the data packets is larger than 65 535 bytes. Whenever a network interface is flooded with such data packets, it would cause the network interface either to be reset or to halt.

The only limitation on a ping-of-death attack is the network speed of the attacker's network interface. If, for example, an attacker is only on a 1 megabyte (MB) connection and this attacker is trying to flood a victim that is on a 10 MB connection or more, it would be technically impossible to flood the victim. The attacker would require a higher connection speed than that of the victim. In the reverse scenario where the attacker is on a 10 MB connection and the victim on a 1 MB connection, the ping-of-death attack would be very successful (Templeton & Levitt, 2000; Karig & Lee, 2001).

This simply shows how easy it is to launch a ping-of-death attack, because so very limited networking knowledge is required (Pfleeger & Pfleeger, 2006; Wang & Stolfo, 2004). Certain intrusion detection systems and firewalls are indeed able to detect these types of attacks (Venter, 2003; McHugh, Christie & Allen, 2000; Debar, Dacier & Wespi, 2000; Deal, 2004). However, in a WSN environment it would be too resource intensive to deploy an intrusion detection system or a firewall solution on the motes, considering the very limited resources that these motes currently possess. The next attack to be discussed, the smurf attack, is a slight variation of a ping-of-death attack.

3.2.2 Smurf

The smurf attack only differs in the way the source of the attack is masked and in the number of victims on a single attack (Lau, Rubin, Smith & Trajovic, 2000; Gouda & Liu, 2005). It is a specific type of a spoofing attack (Mirkovic & Reiher, 2004). The smurf attack firstly alters the source address of the ping packet so that it would seem to the other machines on the network that the attack originated from the victim's own machine (Lau et al., 2000). This is done by spoofing the source address of the ping packet to be the address of the victim's machine.

Secondly, all the bits of the last byte of the destination address are set to all ones (Lau et al., 2000). This would change any destination address which is in the form of 192.168.0.XXX to become 192.168.0.255. The first three bytes of the destination address are used to determine the network subnet and the last byte, in this example, is used to determine the exact machine. This is only the case if the network's subnet mask is 255.255.255.0, as the broadcast address would be different for different subnet masks (Mogul & Postel, 1985). Whenever all the bits of the last byte of the destination address have been changed to all ones (if the subnet mask is 255.255.255.0), the ping request is sent in broadcast mode. This would cause the ping request to be received by all the hosts on the network. Since all the machines that have received the ping request are required to respond to it, they will in essence flood the victim's network interface card.

The echo-charge attack also makes use of manipulating the source and destination headers. This type of attack is briefly discussed in the next subsection.

3.2.3 Echo-Chargen

The echo-charge attack is performed between two hosts. The charge protocol is used to generate a stream of packets between two machines in order to test the network's bandwidth capacity (Garber, 2000; Mukkamala & Sung, 2004).

The following example demonstrates how the legitimate charge protocol would be used. Host **A** would set up a charge process that would generate a stream of echo packets and send them towards host **B**. Host **B** would then receive them and echo all of the packets back to Host **A**. This would then cause host **A** to respond back to host **B** with the same data packets again. The process will continue and seemingly put the network infrastructure of both host **A** and host **B** into an endless loop where they are constantly replying to each other's messages. The endless loop will continue until the user stops the process. The communication between the two devices will afterwards be examined in order to determine if all the packets were continuously received on both ends. This legitimate test will continue until the maximum bandwidth capacity between the two machines has been determined.

In a malicious attempt, the attacker would be host **A**. The attacker would set both the destination and source address of the echo packets in the charge process to that of the same host, i.e. host **B**. This will cause host **B** to hang in a loop as it would constantly create echo messages "sent" to itself and respond to such messages as well.

All three of the attacks discussed above were very straightforward flooding attacks, which only made use of the ICMP protocol. The next subsection discusses one of the more popular denial-of-service attacks, the SYN flood attack.

3.2.4 SYN Flood

The SYN flood attack uses the TCP protocol suite, which makes this attack a session-orientated type of an attack (Harris & Hunt, 1999). One can see it as a connection-orientated attack, because the SYN request is used to establish a connection between two devices. In order for networks to communicate using a TCP connection, a network connection (also referred to as a network session) has to be established (Harris & Hunt, 1999). An example of when a network session is established, is when telnet or SSH (secure shell) is used. In the case where the UDP (user datagram protocol) protocol is used, it is not necessary to establish a connection first for communication to take place (Gehlen, Aijaz & Walke, 2006).

A session is established by means of a three-way connection handshake (Ford, Srisuresh & Kegel, 2005). This three-way connection handshake is depicted in Figure 3.1 (Pfleeger & Pfleeger, 2006).

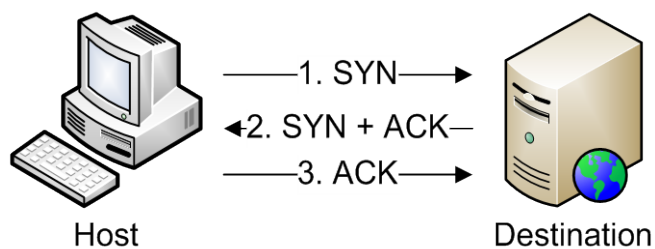


Figure 3.1. Three-Way Connection Handshake

A TCP packet consists of a number of flag bits. Two of these flag bits are denoted the SYN and ACK bits. The process of establishing a session takes place as follows: Firstly the host machine sends a TCP packet with the SYN bit set to 'on'. The destination machine responds with a TCP packet in which both the SYN bit and the ACK bit are set to 'on'. On receipt of this TCP packet from the destination machine by the host machine, the host machine will respond with a TCP packet in which only the ACK bit is 'on'. This process is also illustrated in Figure 3.1.

In an ordinary IEEE 802.11 wireless connection, transmission errors are quite likely to occur as wireless connections are prone to possible interference from other wireless signals. Any machine using the TCP protocol will also have a queue called the SYN_RECV queue. This is to make provision for when packets are lost in the network – thus allowing the machine to have a list of items for which the SYN-ACK messages have been sent, but the corresponding ACK has not yet been received. The SYN_RECV queue is normally flushed within a fairly short time, as the three-way connection handshake is a very quick process (Kim, Choi, Kim & Hong, 2008). The SYN_RECV queue has a limited capacity and under normal circumstances a SYN ACK request would be completed within seconds (Lemon, 2002). If an item is in the SYN_RECV list for some time it will eventually expire and also be flushed from the SYN_RECV queue.

A SYN flood attack occurs when an attacker exploits this three-way connection handshake. It takes place when the attacker sends multiple SYN requests, but never responds with ACK requests. This will cause the victim's SYN_RECV queue to be completely filled up at some stage and thus this machine would not be able to respond to any other SYN requests. As it may take several minutes for items in the SYN_RECV queue to time out and be flushed, it would be very easy for an attacker to maintain the DoS attack over a period of time as a SYN packet would only need to be sent every few seconds (Pfleeger & Pfleeger, 2006).

One can now see that several flooding attacks can be used to cause a DoS attack on machines. This clearly demonstrates that flooding attacks are common and are a great threat to any network environment.

The next section focuses more on *how* flooding attacks occur in a wireless sensor network environment.

3.3 Wireless Sensor Network-Specific Flooding Attacks

As wireless sensor networks always communicate in a broadcast fashion, flooding attacks are much easier to launch as a DoS attack against these devices. Wireless sensor network motes are by default set to accept any incoming traffic. The WSN takes this information and then processes it in order to determine what should be done with it. The receiving and processing of the data obviously takes up valuable battery power from the mote.

If a hostile entity injects a hostile mote into the sensor field of the legitimate WSN, a flooding attack can be launched on the legitimate WSN (i.e. an attack that constantly sends out data to all the motes in its vicinity). This data would normally be a message that polls the motes to see if they are active. The attack would drain the motes in the legitimate WSN of their battery power. This would render the motes useless and they, or perhaps just their power source, would have to be replaced.

To launch a flooding attack in a WSN environment would be even easier than in a normal TCP/IP network. This is because the destination address of each message would already be set to “broadcast to all motes”, due to the fact that broadcasting is the nature in which WSNs operate. All of the earlier attacks could be launched at WSNs. Only slight modifications to the way in which the attacks operate within a TCP/IP network would be necessary, as one would not need to alter the destination address to send messages to all the motes in the vicinity.

Flooding attacks are not attacks that can simply be ignored by users of a wireless sensor network. A continuous check would have to be made on whether the motes in a WSN are being flooded by a hostile network, as such an attack could nullify the purpose of the WSN. Implementing a counter on each mote and specifying a threshold for the number of polls it is allowed to receive per time interval is a possible solution that has been proposed in an attempt to counter flooding (Tseng, Ni & Shih, 2003).

Although the counter system is able to help counter flooding attacks, it is not efficient as ultimately it requires processing on the motes, which in turn drains the battery. Hence it has been decided that it would be a better approach to simply attempt to detect where in the network the flooding mote is, rather than to try and thwart the flooding attack. The reason for this is that the

researcher believes that the only way to really counter a flooding attack in a WSN environment is to physically find and eliminate the rogue mote that is flooding the network.

One possible way to determine if flooding is occurring in the network is by determining if there are sudden influxes of data packets at the base stations in the WSN. The flooding mote will cause more data to be sent from the motes that reside in the area that is being flooded. This influx of data packets can then be dynamically picked up by the base station, if it is able to determine a threshold for how many packets it is supposed to receive per time interval. This method of flooding detection is discussed in detail in Section 9.3. Once it has been determined that a flooding attack is in progress, one can physically examine the area where the flooding is assumed to originate from and eliminate the flooding mote.

3.4 Conclusion

It is evident that flooding attacks in WSNs are a major concern. The basic communication methods in WSNs can almost be seen as flooding, due to the nature of the broadcasted communication. Also, due to the limited battery life of WSNs, a flooding attack would cause much more harm to a WSN than to an ordinary network not powered by batteries.

Because flooding attacks are such a huge concern, the researcher was motivated to solve this problem while trying to also solve the main problem of digital forensic readiness. Since the latter needs to be considered in more detail, the next chapter provides an overview of digital forensics and subsequently digital forensic readiness.

Chapter 4 **Digital Forensics and Digital Forensic Readiness**

4.1 Introduction

Digital forensics is a relatively new science. According to Palmer (2002), the term ‘digital forensics’ is also used interchangeably with the terms ‘computer forensics’ or ‘cyber forensics’. For the purpose of this dissertation, however, digital forensics is often used as an umbrella term, and therefore computer forensics is a subset involving only computers, while cyber forensics specifically involves web-based forensics. Digital forensics stems from the traditional science of forensics, which is not a new discipline at all, but has developed together with the physical and biological sciences over the past number of decades (Palmer, 2002).

The next section defines digital forensics in the way it is referred to in this dissertation, after which the digital forensic process is examined to demonstrate where digital forensic readiness is required to solve the problem stated in this dissertation. Once the topic of digital forensics has been covered, the chapter continues by introducing the reader to digital forensic readiness.

4.2 Defining Digital Forensics

There are several definitions of digital forensics in the available literature. It is essentially a means for gathering electronic evidence during an investigation; however, the following definition by Palmer (2002) summarises this science: Digital forensics entails “the use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation, and presentation of digital evidence derived from digital sources for the purpose of facilitation or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations”.

People from several different areas of expertise, e.g. legal advisors, technical experts and traditional forensic experts, may be involved during a digital forensic investigation. This immediately poses a problem because, according to the legal advisers, digital evidence has to be presented in the same way that traditional forensic evidence is presented (Reith, Carr & Gunsch, 2002; Palmer, 2002). Due to the nature and scope of digital evidence, however, it is very difficult to present this evidence in a court of law. Much research currently focuses on this problem (Rogers & Seigfried, 2004; Baryamureeba & Tushabe, 2004; Beebe & Clark, 2004; Oppliger & Rytz, 2003; Yasinsac, Erbacher, Marks, Politt & Sommer, 2003; Mocas, 2004; Leigland & Krings, 2004; Meyers & Rogers, 2004). It was established in the above research efforts that the following requirements as devised by Daubert and Frye (Gianelli & Imwinkelried, 1999) have to hold, before any court of law will accept the digital evidence (Bernstein, 2000; Jonakait, 1993; Carrier, 2002):

- The digital forensic theory or technique must have been reliably tested.
- The digital forensic theory or technique must have been subject to peer review and publication.
- The known or potential error rate of the digital forensic method used should be known.
- The digital forensic theory or method must have been generally accepted by the scientific community.
- An acceptable digital forensic process needs to be followed in acquiring and presenting the digital evidence.

Many research efforts in digital forensics still have to conform to the above requirements due to the fact that the discipline is so young. Nevertheless, this research area is growing fast with many new and interesting applications, as will be presented in the section on digital forensic readiness. As indicated above, the digital forensic process to be followed during a digital forensic investigation is paramount to the credibility of such an investigation. The next section is aimed at familiarising the reader with the digital forensic process and the steps in a digital forensic investigation.

4.3 Digital Forensic Process

Digital forensics should be seen as a precise science. One could make this observation as every small step in the entire process should follow a list of pre-setup rules, laws and procedures. Even the smallest of errors in the process could lead to the evidence becoming worthless in a court of law. Thus it is important to ensure that every step of the digital forensics process is well documented and understood. For this reason, a digital forensic process model was developed. The basic digital forensic process model, as found in most of the literature, is defined in Baryamureeba and Tushabe (2004) and is briefly summarised below:

- **Collection** of digital evidence, which involves the search, recognition, collection, and documentation of the digital evidence. It is also paramount to prove in a court of law that the digital evidence has not been tampered with or changed, either accidentally or intentionally.
- **Examination** of the digital evidence, which involves revealing of hidden and obscured information while ensuring that the digital evidence can still be visibly presented and that one is able to still explain the origin and significance of the digital evidence.
- **Analysis** of digital evidence, which involves looking at the result of the examination in order to determine its significance and investigative value to the case.

- **Reporting** of digital evidence, which entails writing a report in order to outline the examination process and relevant data recovered from the overall digital forensic investigation.

This basic digital forensic process model is expanded and refined in much more detail in Baryamureeba and Tushabe (2004), Pollitt (2007), Reith, Carr and Gunsch (2002), Baryamureeba and Tushabe (2004). However, this study does not focus on the digital forensic investigation process as such, because digital forensic readiness (the focus of this dissertation) only forms part of the collection process within digital forensics (Carrier & Spafford, 2003).

As has now been pointed out, digital forensics is a very precise process and quite a costly exercise to perform. One should take into consideration every little step of the digital forensic process, and all of these steps should be very well document. A single mistake would cause the evidence to be inadmissible in court. Digital forensics could be a very timely and costly process, and in order to minimise possible mistakes, the reader is now introduced to digital forensic readiness.

4.4 Digital Forensic Readiness

Digital forensic readiness is a subsection of digital forensics. Digital forensic readiness more specifically forms part of the first subprocess of the digital forensics process, namely collection.

As the goal of this study is to achieve digital forensic readiness in a wireless sensor network environment, the following subsections define digital forensic readiness and show how digital forensic readiness can be achieved.

4.4.1 Defining Digital Forensic Readiness

Before trying to achieve digital forensic readiness in any type of environment, it is essential to establish an acceptable definition for it. However, since digital forensic readiness is still a fairly new concept, many people have different opinions about it.

Tan (2001) identifies two objectives as part of a definition for digital forensic readiness that have to be carefully balanced: maximising the ability to collect credible digital evidence, as against minimising the cost of performing a digital forensic investigation. Tan also argues that several steps need to be taken to ensure that an environment is ready as far as digital forensics is concerned. Rowlingson (2004), on the other hand, suggests ten steps that describe the key activities in implementing a digital forensic readiness program. Because Rowlingson's steps have actually been designed to create a business process model for digital forensic readiness, this dissertation gives

preference to Tan's two objectives for meeting the requirements of digital forensic readiness in a WSN environment.

Even though Tan's objectives provide a very good definition of digital forensic readiness, it is still important to refine them to be more specific to a WSN environment. For the purpose of this dissertation, digital forensic readiness is defined as *the notion to perform a digital forensic investigation in the shortest amount of time with the least amount of cost and without having to disrupt the original network that has to perform mission critical tasks*. This definition is set as the main goal for achieving digital forensic readiness on WSNs.

The following section analyses and discusses the definition of digital forensic readiness as it has been proposed in this dissertation.

4.4.2 Achieving Digital Forensic Readiness

The definition provided in the previous section focused on three elements: the time period required to perform a digital forensic investigation; the cost involved to perform a digital forensic investigation, and the ability to collect the evidence without disrupting the environment.

Each of these three elements is discussed in the following subsections as they are pivotal in achieving digital forensic readiness.

4.4.2.1 Time Period Required to Perform a Digital Forensic Investigation

Digital forensic readiness is put in place to decrease the time period necessary to perform a digital forensic investigation. Digital forensic readiness focuses on the collection part of the digital forensic process.

In an environment where digital forensic readiness is implemented, the time it takes from when the incident occurs up to the time it takes to where the incident-related information can be analysed, should be kept to a minimum. This is because the digital forensic readiness systems ensure that the information is captured into a separate environment on which the work-related systems are not dependent.

The nature of the digital forensic readiness environment being separate from the work-related environment brings us to the next important factor, namely the cost involved to perform a digital forensic investigation.

4.4.2.2 Cost Involved to Perform a Digital Forensic Investigation

Approaching any organisation with the idea of implementing a digital forensic readiness environment brings up the ever-looming question of the cost related to it. Although this is really a

costly endeavour, one should look ahead at the costs involved in doing a digital forensic investigation without any precautions having been taken. Implementing a digital forensic readiness environment is costly because the needs of each organisation are different and the digital forensic readiness solution might need to be custom fitted to the organisation (Rowlingson, 2004). In operation, this solution would also need to be independent of the daily tasks of the organisation, as the digital forensic readiness solution should be able to form part of a digital forensic investigation if the need arises (Rowlingson, 2004).

In the event that a digital forensic investigation should be launched in an environment that is not complaint to a digital forensic readiness solution, this would cost an organisation a lot of money. One should realise that if an incident occurred on the important servers of an organisation, it may sometimes be required to power down those important servers in order to acquire a forensic image of the system. When an organisation is unable to continue with its daily tasks, it could lose a lot of money (Zhang, Zhang & Wang, 2010). Also, in most cases one would not be allowed to take mission critical systems, i.e. the bank's mainframe server, offline if they are going to lose so much – then you have no option but to perform live forensics.

Considering the cost of implementing digital forensic readiness versus the cost involved in conducting a digital forensic investigation without having implemented digital forensic readiness, it would be fair to say that one would rather be safe than sorry. One would rather incur the costs of digital forensic readiness and be ready for an incident, than having to fork out more money in the case that an incident does occur (Tan, 2001; Grobler, Louwrens & von Solms, 2010).

This subsection briefly touched on the cost involved to perform a digital forensic investigation in an environment where digital forensic readiness has not been implemented. The next subsection looks at how the collection of evidence occurs in a digital forensic readiness environment.

4.4.2.3 Collecting Evidence without Disrupting the Environment

The collection of evidence in a digital forensic readiness environment would be much easier than in an environment without digital forensic readiness. This is due to the nature of the digital forensic readiness environment, which is separated from the mission critical environment.

One would be able to take down an entire digital forensic readiness system if it has been configured to run as a system separate from the main environment. This allows one to perform a thorough collection of digital evidence without being pressed for time. In turn, it also allows one to

be more precise in the digital forensic process, as there would not be pressure from the organisation itself to perform the collection in a rushed manner.

Considering the time, cost and the disruption to the environment, this study has shown that a digital forensic readiness environment would be highly beneficial to any organisation. The next section concludes the chapter and provides final remarks on this part of the dissertation.

4.5 Conclusion

This chapter first introduced the reader to the umbrella term of digital forensics. Digital forensics was shown to be an extremely precise process and quite a costly exercise to perform. Every step of the digital forensic process should be very well documented, as a single mistake would cause the evidence to be inadmissible in court. Chapter 4 provided an overview of digital forensic readiness in an attempt to warn against the cost, time and disruption involved in a possible digital forensic investigation. Digital forensic readiness was for the purpose of this dissertation defined as the notion to perform a digital forensic investigation in the shortest amount of time, with the least amount of cost, and without having to disrupt a system under investigation that has to perform mission critical tasks.

Now the reader has been familiarised with all of the necessary background topics that will be focused on in this dissertation. The next part proposes a model that contains all the requirements to be met in order to achieve digital forensic readiness in a WSN environment.

Part III: Model

Chapter 5 Model requirements

5.1 Introduction

Part II of this dissertation provided the required background knowledge needed to proceed with the research. This background information puts one in the position to understand the problem that this study focuses on: *there is no ready solution to apply digital forensics in a WSN without infringing on the setup, functionality and performance of the WSN*. On the basis of how WSNs operate and communicate in the field, one can immediately imagine that security on these devices is an important factor. Unfortunately, due to the broadcast fashion of communication within WSNs, security is a fairly difficult feature to implement for WSNs. In broadcasted communication, data packets can be easily intercepted by an attacker, as the data packets are by default transmitted to all the nodes in the nearby vicinity.

In previous research, the researcher suggested a security layer for WSNs (Mouton & Venter, 2009). To his knowledge, the only real way to add a security layer to WSNs would be to modify the devices that have already been deployed in the field (Mouton & Venter, 2009). The security protocol, as proposed by Mouton and Venter (2009), relies on secure data channels between the nodes and each node is allocated a pre-assigned security key before initial deployment into the field. These security keys are then used to encrypt data packets between the nodes. However, the security keys are changed on an incremental basis after each transmission in order to make it more difficult for an attacker to crack the security codes. Other authors have made the same findings as the author of this dissertation, as all their approaches state that security should be implemented and devices should be configured before initial deployment of the devices (Karlof & Wagner, 2003; Perrig et al., 2001; Zhu, Setia & Jajodia, 2006). However, in this dissertation the assumption is made that the devices have already been deployed into the field and cannot be retracted to be reconfigured and redeployed, as this will be a costly and time-consuming process.

Due to the above assumptions and reasoning, the researcher decided that it would be more cost-effective and time-effective to propose a concept in which security is implemented on an independent secondary WSN (where this secondary WSN is introduced to the same sensor field as the original WSN). For this very reason it was decided to move towards the field of digital forensics in an attempt to add a digital forensic readiness layer to an existing WSN environment by means of adding an independent secondary forensic WSN. Adding a digital forensic readiness layer to the WSN environment has a number of advantages. A digital forensic readiness layer could be used for the obvious purpose of digital forensics, but it can also be used to monitor network patterns within a WSN environment or even to gather statistics about the existing WSN that it is forensically

monitoring. From here onwards, the researcher will address all the considerations that were taken into account in designing a digital forensic readiness layer for a WSN environment.

Firstly, before proposing a model, one should focus on what unique requirements WSNs have in terms of digital forensic readiness. WSNs have special needs in terms of digital forensic readiness when compared to ordinary WLANs and, hence, have more specialised requirements than would be found in ordinary wireless networks (also known as wireless local area networks or WLANs). These will be examined in depth in the section to follow. WLAN environments function on the TCP/IP protocol stack, which provides an easier means of validating the authenticity and integrity of data packets that have been captured. WLANs also do not have the power supply constraints that WSNs have, just to mention one of the many other differences between a WSN and a WLAN.

The remainder of this chapter is devoted to examining all of the differences between WSNs and WLANs in terms of digital forensic readiness. The researcher firstly addresses how WSNs differ from WLANs and then discusses the special requirements of WSNs. Section 5.3 summarises section 5.2 into a table, which contains a list of factors to adhere to when considering to implement digital forensic readiness in a WSN environment. Section 5.4 concludes the chapter.

5.2 Special WSN requirements

WSNs have more specialised needs when compared to IEEE 802.11x WLANs and, hence, have more specialised requirements than would apply to WLANs. There are many important factors that make a WSN unique and distinguish it from a WLAN. The factors that are addressed in this dissertation are the following:

- Communication protocol
- Proof of authenticity and integrity
- Time stamping
- Modification of the network after deployment
- Protocol data packets
- Radio frequencies
- Power supply
- Network overhead
- Data interference

The factors listed above are the main ones that differentiate WSN environments from WLAN environments. The reasoning behind the choice of these factors will become apparent in the subsections to follow, where each factor is addressed individually. It is, however, important to remember that the core of the argument about the importance of these factors concerns the manner in which they influence the design decision of how to implement a digital forensic readiness application for WSNs.

While examining each of these factors, it is important to note that the researcher assumes that no modification to the original WSN (hence forward referred to as oWSN) is allowed and thus a secondary independent forensic WSN (hence forward referred to as fWSN) would be used for the digital forensic readiness implementation of the oWSN. Figure 5.1 has been included here to show an example of an oWSN with an overlaying fWSN.

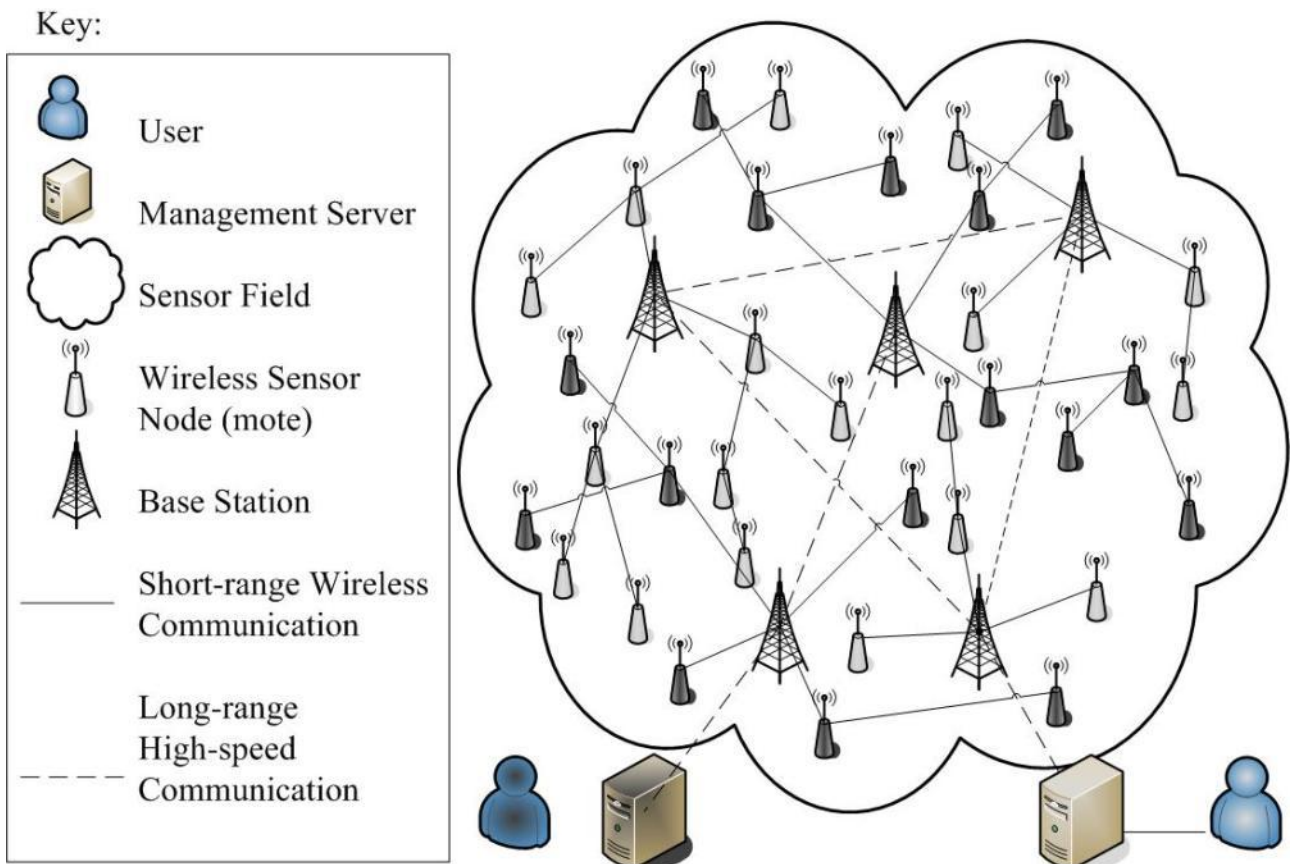


Figure 5.1. A graphical representation of an oWSN with an overlaying forensic WSN

In Figure 5.1, the darker-shaded equipment depicts the fWSN and the lighter-shaded equipment depicts the oWSN. This figure is used again later on in this dissertation. However, it has been inserted here to provide the reader with a better understanding of an oWSN with an overlaying

fWSN. The discussions in each subsection below briefly focus on how these factors differ between a WLAN and WSN, and subsequently the focus shifts to addressing them in WSNs.

5.2.1 Communication Protocol

All communication within a WSN occurs in a broadcast fashion (Akyildiz et al., 2002; Tseng, Ni & Shih, 2003). The default functioning of a mote in the sensor field is to receive all packets. Upon receipt of a packet, the mote then has to analyse if the packet was meant for it or not. This analysis requires some processing that drains the battery of the mote, which is an important consideration in WSN communication.

The broadcasting technique used in WSNs is very different from the communication techniques used in an IEEE 802.11x wireless network. In the WLAN environment, one can determine if a packet has arrived at its destination by monitoring the network, since acknowledgement packets are sent to confirm the receipt of packets (Xylomenos & Polyzos, 1999; Xylomenos, Polyzos, Mahonen & Saaranen, 2001). This is not the case in a WSN environment.

Due to the broadcasting fashion in which WSNs communicate, the mote that broadcasts packets will never be completely sure whether the packet was received by the mote for which the packet was intended. This uncertainty could be overcome by introducing a communication protocol that allows the receiving mote to reply with a receipt acknowledgement packet. However, because this would require extra transmissions that can lead to faster battery drain, this procedure cannot simply be implemented in all WSNs. Using a communication protocol that has receipt acknowledgement packets also has several other disadvantages. If a flooding attack is launched against the oWSN, it would compel the oWSN to reply to each flooding attempt with receipt acknowledgement messages, which would then flood the entire oWSN.

Considering that a protocol founded on receipt acknowledgement packets can have such a severe impact on a WSN environment, it seems quite impractical to use such a protocol in this environment. Hence the researcher agreed to accept that most WSN motes will be uncertain as to whether or not packets have actually arrived at their destination. This causes severe problems in terms of digital forensic monitoring with a secondary network. The packets received by the oWSN base station might differ from those received by the fWSN base station if some of the packets are lost or transmission errors occurred in either of the two WSNs. Transmission errors can range from two motes sending messages at the same time and the messages colliding, or weather conditions influencing the wireless range of the motes. In the case of the fWSN, however, this problem could be avoided by implementing a protocol that uses receipt acknowledgement packets, because it is a requirement of a forensic network to always assure the authenticity and integrity of the information

received at either point of the communication line. In order to achieve sound digital forensic readiness, it is crucial to provide such authenticity and integrity.

The next subsection focuses on defining what the researcher perceives as such authenticity and integrity. The differences between maintaining the authenticity and integrity from a WLAN and a WSN perspective are also discussed, as well as possible ways of maintaining authenticity and integrity within a WSN environment.

5.2.2 Proof of Authenticity and Integrity

Authenticity and integrity first need to be defined as there could be different opinions on precisely what each of them means. Authenticity is derived from the word authentic, which is defined as “of undisputed origin and not a copy” (Oxford English Dictionary, 1989). In the context of this dissertation, authenticity is defined as *the certainty that the origin and destination of the data packet are kept intact throughout its whole lifetime*. The lifetime of a data packet runs from the time that it is sent from the first mote up to the time when it is received and processed by the base station. Next, integrity is defined as “internal consistency or lack of corruption in electronic data” (Oxford English Dictionary, 1989). In the context of this dissertation, integrity is defined as *the certainty that the correctness of the data within the data packet is kept intact throughout the lifetime of the data packet*.

Numerous techniques for proving the authenticity and integrity of packets in WLANs have already been published (Chen, Jiang & Liu, 2005; Komori & Saito, 2004; Guizani & Raju, 2005). Firewalls, Intrusion Detection Systems, Wireless Routers and Wireless Network Interface Cards are all examples of equipment you would find in WLANs and most of these devices have the ability to generate a log or some other way of showing which data packets have passed through the network. Most of these abilities are fairly simple techniques that are performed by the device itself. In most cases where a log file is generated, it is safe to assume that the information reflected in the log file is actually the true pattern of traffic that has passed through the device. However, this is only the case when it is certain that the device is not defective or that the log file has not been tampered with. One can confirm the authenticity and integrity of this single log file by comparing it to the other log files of the devices through which this single packet has travelled, as most devices in a WLAN environment should have some form of logging. In a WSN environment, however, very little or no logging is performed on the motes in the sensor field, due to various reasons. These reasons can include the limited power source and the limited storage space that the devices in a WSN have. WSN equipment, by default, only does logging at the base station and if logging were to be required at every mote, one would have to implement this oneself.

Implementing logging software by oneself obviously raises another issue, namely as to the trustworthiness of the code with which one does the logging. Tried-and-tested techniques for logging are generally more trustworthy than one's own attempts at implementing logging. It is easier to defend the authenticity and integrity of a well-known logging technique than that of a self-developed logging technique. In the case where a self-developed technique is used, it must be based on some solid theory as to why it can provide authenticity and integrity. Because WSNs differ so significantly from WLANs, the researcher has decided to propose a form of logging that is based on the Casey Certainty Scale (Casey, 2002).

According to Casey (2002), the integrity and authenticity of information is more certain if this information was recorded by different independent sources. Each mote can, in essence, be seen as an independent source. Thus, the authenticity and integrity of each packet can be determined based on the number of motes in the network that have received the same broadcasted packet. This study therefore assumes that, in accordance with the Casey Certainty Scale (Casey, 2002), a packet that has been seen by a larger number of motes has far greater authenticity and integrity than a packet that has only been seen by a few motes in the fWSN.

Fortunately, in a WSN environment, multiple motes each tend to be able to capture the same data packet simply because they are all in range of a particular broadcasted packet. This is a feature of WSNs, which is not the case in WLANs. Most devices in WLANs will ignore packets that are not meant for them and do not even attempt to log these packets. The opposite is true for WSNs, where motes attempt to capture every data packet within range. This feature of WSNs can be successfully exploited in an attempt to prove the authenticity and integrity of packets in the WSN. All the packets captured by each independent fWSN mote could be forwarded to the base station as a central point of analysis, in an effort to prove the authenticity and integrity of the data packet according to the Casey Certainty Scale (Casey, 2002).

The above technique constitutes only one of several ways to determine the authenticity and integrity of the packets in a WSN. Time stamping and the sequence of packets can also be used for this purpose. However, time stamping in a WSN is a tedious task. The next subsection is nevertheless devoted to it.

5.2.3 Time stamping

Time stamping in a WLAN environment is a fairly easy task, since all the devices in a WLAN would under normal conditions either have access to a time server or be set with the correct time. Thus, time stamping in the logs for a WLAN would under most conditions be correct, provided that the device has not been tampered with or is not faulty. In the case of a WSN, however, only the

management server (which is connected to the base station) has a sense of time. The motes in a WSN environment have no sense of physical world time and the only measurement they can use is their own sense of time, which is the time that has elapsed since they were switched on (Sundararaman, Buy & Kshemkalyani, 2005; Su & Akyildiz, 2005; Sun, Ning & Wang, 2006). Such elapsed time can be measured on WSN devices in terms of ticks, where each tick represents 100 nanoseconds (Sundararaman, Buy & Kshemkalyani, 2005). This uptime, although fairly accurate, is a poor indication of time, because each mote in the entire network has to be switched on simultaneously and the time should also be synchronised by transmitting their uptime along with their data packets. It is impractical to switch on motes simultaneously and synchronisation is not feasible due to resource restrictions.

When tests were conducted concerning the time stamping of WSNs, the researcher noted that it takes at most one second to capture any data packet and transmit it to the fWSN base station. This nevertheless introduced a time delay between capturing a packet and receiving it at the base station. The time delay also differed according to the distance of the fWSN mote from the base station in terms of hops and physical distance. Thus the time stamps at the base station are not an accurate reflection of when the packet was initially captured, as the base station is the only device that can assign an accurate time stamp if it is connected to the management server. (The reason for this is that only the management server has access to a time server (Sundararaman, Buy & Kshemkalyani, 2005; Su & Akyildiz, 2005).) It is also important to note that each fWSN mote captures packets sequentially, in the order that the oWSN motes transmit their data packets. This proves to be a vital piece of information, because one would then be able to claim that even if the time stamps are altered, the sequence would still be intact. The order in which they arrive at the fWSN will also stay intact even if the time stamps are slightly delayed. This allows one to assume that the time delay between capturing the packet and sending it to the fWSN base station would not really affect the authenticity and integrity of the packets, as the sequence of packets can be used to determine their authenticity and integrity.

The trustworthiness of log time stamps is an issue that many digital forensics researchers have queried and investigated (Schatz, Mohay & Clark, 2006; Schneier & Kelsey, 1999). The dilemma faced by the fWSN is merely intensified. It becomes a more severe issue to trust the time stamps, as the limitation of having no access to a centralised time server for WSNs might prevent them from reflecting the correct time. However, since the sequence of the data packets is not altered, this (rather than the time stamps) could be used to verify the authenticity and integrity of the data packets. The researcher therefore assumes that the fixed sequence of the data packets is more important than the precise time at which they were transmitted. In fact, more information can be

gathered by looking at the sequence of the data packets than by looking at their time of transmission.

It is then sufficient to capture the data packets and merely provide a time stamp for them as soon as they arrive at the fWSN base station. In the event that this is done, one would admittedly create a time stamp error. The time stamp error would nonetheless be a constant error for each oWSN mote, as it would reflect the time the data packet was first transmitted together with the added time it took for this data packet to reach the fWSN base station. The fWSN base station, which is connected to a time server, assigns a time stamp to each data packet upon its arrival there. This allows the order of the packets to be kept intact and records a one-second error on the time stamp of each packet due to the fact that the time stamps is assigned by the base station and not by the forensic mote that captured the packet initially. Due to the fact that the time stamp error stays constant for all the packets received from a specific mote in the sensor field, it is still possible to guarantee the authenticity and integrity of a packet by analysing the sequence in which the packets were received. This constant error could be measured, if needed, by comparing the time stamps at the oWSN base station and the fWSN base station. The time stamp, combined with the sequence of the data packets, would then be sufficient to be used in a forensic investigation.

Another issue that the researcher considered while examining the differences between WLANs and WSNs is the feasibility of modifying the network after it has been deployed. This matter is discussed in the following subsection.

5.2.4 Modification of the network after deployment

Being able to modify the network after deployment is the only factor that was found to be fairly similar between WLANs and WSNs, as it is always possible to modify code on a device by retracting it from the sensor field, redeveloping it and then redeploying it back into the sensor field. However, the practicality of altering oWSN devices after deployment must be taken into consideration. It is important to remember that oWSN motes are usually scattered in an area and to alter them, one would have to physically collect the entire oWSN and redeploy it. Hence, it seems essential that the oWSN should not be modified to accommodate an fWSN solution. This is the very reason why the researcher opted to add an overlaying fWSN to the oWSN in order to do all the forensic monitoring. The overlaying fWSN would consist of a separate set of WSN motes that does not affect the oWSN and also requires no modification of the oWSN as shown in Figure 5.1.

The difficulty and impracticality of modifying the oWSN has led the researcher to believe that this should also be seen as a specific requirement when attempting to provide forensic readiness to a WSN environment. Considering that we cannot easily alter the oWSN, we must

ensure that the fWSN should be able to handle any type of protocol headers and footers that could originate from the oWSN. Against this background, the next subsection focuses on the protocol data packets that are used by WSN devices and the reasons why it is important to take this into consideration when implementing an overlaying fWSN.

5.2.5 Protocol Data Packets

The oWSN can have many different types of communication protocols in its normal operation. For example, the data packets can include packets to determine the routing protocol, sensory packets, encrypted packets or even malformed packets. In order to ensure that all of the possible protocols used in WSNs are encapsulated in this approach, it has been assumed that the oWSN uses an address-free protocol. This protocol generates the largest amount of network overhead in WSNs, as it would cause data to be sent from a source mote in the network to every other mote in the network on each data transmission (Dunkels, Osterlind & Zhitao, 2007). The most commonly used address-free protocols are data dissemination protocols, where neither the sender mote nor any of the other motes in the network knows the address of the receiving mote (Dunkels, Osterlind & Zhitao, 2007). If the fWSN is able to successfully log this communication from an address-free protocol in a way that ensures authenticity and integrity, one could assume that the name-based WSN protocols would effortlessly be accounted for, as they have much less network overhead (Dunkels, Osterlind & Zhitao, 2007).

As is also the case in WLANs, the motes in the fWSN should listen in promiscuous mode and be able to handle any type of packet that is transmitted or received by the oWSN. The researcher defines promiscuous mode to be a configuration of the WSN mote in which all traffic within the WSN mote's frequency range and wireless range will be received by the WSN mote. Thus, if an attacker uses a foreign mote to inject data into the oWSN, the fWSN should also be able to listen in on this data. This requirement should be fairly simple to adhere to, because if the fWSN is implemented on the same type of equipment, it should be possible to intercept all communication.

Lastly, the fWSN should be using a name-based WSN protocol for communication between other fWSN motes, as it is more optimal in terms of network overhead than address-free protocols. In name-based protocols the source mote knows the address of the receiving mote and the motes between the sender and receiver know the path to the receiving mote (Dunkels, Osterlind & Zhitao, 2007).

Considering that we cannot alter the oWSN, we should be able to ensure that the frequency that is used to communicate is the same range as one which the fWSN can forensically monitor.

Thus, the next subsection will focus on the radio frequencies that are used by WSN devices and why it is important to take this into consideration when implementing an overlaying fWSN.

5.2.6 Radio Frequencies

Both WLAN and WSN devices communicate on specific frequencies. The WLAN devices communicate on either the 2.4 GHz ISM (Industrial, Scientific and Medical) Band or on the 5 GHz ISM Band (Crossbow Technology Inc, 2007). Any devices adhering to either the IEEE 802.11a standard or the IEEE 802.11n standard are capable of utilising the 5 GHz ISM Band. Devices adhering to either the IEEE 802.11b standard, IEEE 802.11g standard or the IEEE 802.11n standard are capable of communicating on the 2.4 GHz ISM Band (Doufexi, Armour, Butler, Nix, Bull, McGeehan & Karlsson, 2002). WSNs adhere to the IEEE 802.15.4 standard, which condones communication on the 2.4 GHz up to 2.4835 GHz band.

There are also other frequencies that can be utilised by WSN devices and different WSN devices can use different stepping on the 2.4 GHz frequency. Stepping refers to the WSN devices being able to communicate in different spectrums of the 2.4GHz frequency with a very small frequency difference between the two spectrums. It is therefore an important requirement that the fWSN is implemented on either the same type of WSN equipment used for the oWSN or equipment that is capable of tuning into the same frequencies utilised by the oWSN.

This is almost a given requirement as most organisations will tend to use the same type of equipment for both WSNs. It would simply be the most practical thing to do and cause the least difficulties in tuning to the correct frequencies and utilising the same kind of processing. As transmitting packets throughout the network is one of the main factors that drain power and thus decrease the lifetime of the WSN, we will focus on power consumption in the next subsection.

5.2.7 Power Supply

As mentioned earlier, WSN equipment uses an external power supply that consists of some type of battery pack. This is a huge difference between WSN equipment and WLAN equipment. Most WLAN equipment is either powered by the motherboard it is plugged into or it is plugged into a wall socket that provides a constant power supply. The WSN equipment has to rely on battery packs for power, because they are mostly scattered in open fields where no power sources are nearby. A lot of research is currently being done on how to decrease power consumption in WSN devices or how to use something like solar energy (Ye, Heidemann & Estrin, 2002; Slijepcevic & Potkonjak, 2001; Wander et al., 2005; Polastre, Hill & Culler, 2004).

Until the power consumption issue has been resolved, it is important to take this requirement into consideration. Any protocol or model suggested for a WSN should use minimal amounts of processing power and have as little as possible radio transmissions between the motes. These are the two main elements that drain the battery packs (Shnayder, Hempstead, Chen, Allen & Welsh, 2004).

While considering the power consumption factor, one should also ensure that no fWSN adds any overhead to the oWSN. This is because any overhead to the oWSN would increase the power consumption of the oWSN and thus decrease its lifetime. The following subsection focuses on the requirement that the fWSN should not increase the network overhead of the oWSN.

5.2.8 Network overhead

Network overhead is quite an important factor when dealing with WSN devices. An increase in network overhead would cause a decrease in battery lifetime for the entire WSN. As discussed earlier, WSN devices send messages in a broadcast fashion. This has the disadvantage that all the motes in the oWSN need to listen to any incoming packets. Thus, if the fWSN were to send out any data packets, the oWSN motes nearby would be able to receive them. The oWSN would then need to analyse each packet to determine if it was addressed to them and if not, they will have to drop the packet.

The process of receiving and analysing packets mentioned above already introduces a severe overhead into the oWSN. Thus it would require the fWSN to communicate on a different radio frequency as the one on which it listens for data packets from the oWSN. Data packets sent on a different frequency cannot be seen by motes in the oWSN and will therefore not cause any overhead in the network.

If the communication from the fWSN is on a different frequency, we also have the added benefit that the fWSN will not have any effect on the data received by the oWSN. We can also see it as a requirement that the fWSN should not be able to alter any data received by the oWSN. This is briefly discussed in the following subsection.

5.2.9 Data Interference

The fWSN should not be able to influence any sensory data, for example temperature or humidity readings, which have been received or are sent by the oWSN. Thus, as referred to in the previous subsection, the fWSN is only able to transmit data on a frequency that is not in range of any device in the oWSN. This is done deliberately to ensure that there is no conflict between the communication on the oWSN and fWSN. The data packets sent on this different frequency will

never be able to reach any oWSN device, thus one need not be concerned about the issue of traffic from the fWSN influencing any data from the oWSN.

The data transmitted from the oWSN can come in many different forms. The type of data packets within an oWSN are all dependent on the type of communication protocol used. Some of these communication protocols may even use a type of encryption to keep the channels of communication more secure. For these reasons it must be ensured that the fWSN is not able to influence any of the data packets that are transmitted within the oWSN, as this can cause errors between the transmissions of the oWSN nodes.

All the major differences between WSNs and WLANs have now been discussed. The following section is devoted to arranging all these factors into a single list of requirements that need to be adhered to when implementing digital forensic readiness in a WSN environment.

5.3 Forensic Readiness Requirements for WSNs

The previous sections identified the factors that differentiate between WLANs and WSNs in terms of digital forensic readiness. These factors were simply broad overviews of issues to be considered in the WSN environment (most of which do not exist in a WLAN environment).

The researcher subsequently proposes a broad, yet detailed set of the important requirements to be adhered to in order to successfully implement digital forensic readiness in a WSN environment. This list of requirements (see Table 5.1) could serve as a good starting point for anyone working on digital forensic readiness and makes it easier for an individual to implement digital forensic readiness within a WSN environment. Most other researchers focus mainly on one or two of these requirements by going into more detail on them in their research papers, but many other requirements are usually not mentioned, regardless of their importance.

Table 5.1 gives a brief but comprehensive overview and summary of all the important requirements that need to be taken into account in order to achieve digital forensic readiness in an IEEE 802.15.4 WSN environment (as discussed in detail in the previous section).

Table 5.1. Factors to take into consideration when implementing digital forensic readiness on an IEEE 802.15.4 wireless sensor network

Factors	Detailed requirement list
Communication Protocol	1. The fWSN should use a receipt acknowledgement packet protocol to ensure that all data packets captured by the motes in the field do indeed reach the base station.
	2. The broadcasted communication from the oWSN should be intercepted in a manner that ensures that the data packets are not altered in any fashion.
	3. The fWSN should be able to capture all possible types of communication that can be sent from the oWSN.
Proof of Authenticity and Integrity	4. The authenticity and integrity of all the data packets should remain intact while being captured on the fWSN.
	5. The data packets that are captured in the fWSN should be stored in such a way that their authenticity and integrity are not compromised.
	6. It should be possible to verify the authenticity and integrity of all the data packets in case a digital investigation takes place.
Time Stamping	7. The data packets should have a time stamp assigned to them that does not violate their authenticity and integrity.
	8. The sequence of the packets captured should reflect the true sequence in which they were transmitted from the original network.
Modification of the network after deployment	9. It should be possible to implement the fWSN without any modification of the oWSN.
Protocol Data Packets	10. The fWSN should be designed in such a manner that the network topology or the routing protocol used by the oWSN does not influence the fWSN's operation.
Radio Frequencies	11. The fWSN should be able to communicate on the same radio frequencies that are available to the oWSN.
	12. All communication within the fWSN should occur on a frequency not utilised in the oWSN.
	13. If an intruder WSN is in the area and communicates on a frequency that influences the oWSN, then the fWSN should be able to forensically capture these data packets.
Power Supply	14. The fWSN should not increase power consumption in the oWSN and the fWSN should have at least the same or a longer network lifetime than the oWSN in terms of battery power.
Network Overhead	15. While intercepting communication, there should be no extra network overhead on the oWSN.
Data Interference	16. The fWSN should by no means be able to influence the oWSN or influence any sensory data transmitted within the oWSN.

This list provides a sound basis to start from when attempting to achieve digital forensic readiness in a WSN environment.

The following section concludes this chapter by providing a brief overview of the information contained in it.

5.4 Conclusion

This chapter was used to identify some crucial differences between WSN environments and WLAN environments. The researcher used a scenario where two networks must co-exist in the same environment to determine the differences. The differences can be seen in the following list:

- Communication protocol
- Proof of authenticity and integrity
- Time stamping
- Modification of the network after deployment
- Protocol data packets
- Radio frequencies
- Power supply
- Network overhead
- Data interference

This list of differences allowed the researcher to compile a more specific set of factors to take into account when designing a digital forensic readiness model for WSN environments. These factors were designed almost as a set of rules that can be used as a checklist to determine if the proposed model will indeed work. Such a list could provide anyone who would further want to attempt anything along the line of digital forensic readiness within WSN environments with a good starting point.

Now that we have a starting point, the next chapter will propose a model to implement digital forensic readiness in a WSN environment.

Chapter 6 **Proposed Model**

6.1 Introduction

The previous chapter proposed a set of factors that need to be taken into account while proposing a digital forensic readiness model for WSN environments. These requirements are very specific to IEEE 802.15.4 WSNs.

The following section (6.2) is used to propose a model on how to achieve digital forensic readiness in a WSN environment. Firstly, it will graphically represent the deployment of such a model in any sensor field and also discuss the details on how the model would be used. It is important to note that this chapter only provides a broad overview of the inner working of the implementation. A detailed discussion follows in a later chapter.

The model having been proposed, Section 6.3 shows how and why the model adheres to all the factors that were specified in the previous chapter. It is shown how the model adheres to each of the proposed factors.

6.2 Model

In the previous chapter, the researcher already mentioned that the model would include the existing WSN with an overlaying fWSN to provide the digital forensic readiness layer. Figure 6.1 graphically depicts an oWSN with an overlaying fWSN.

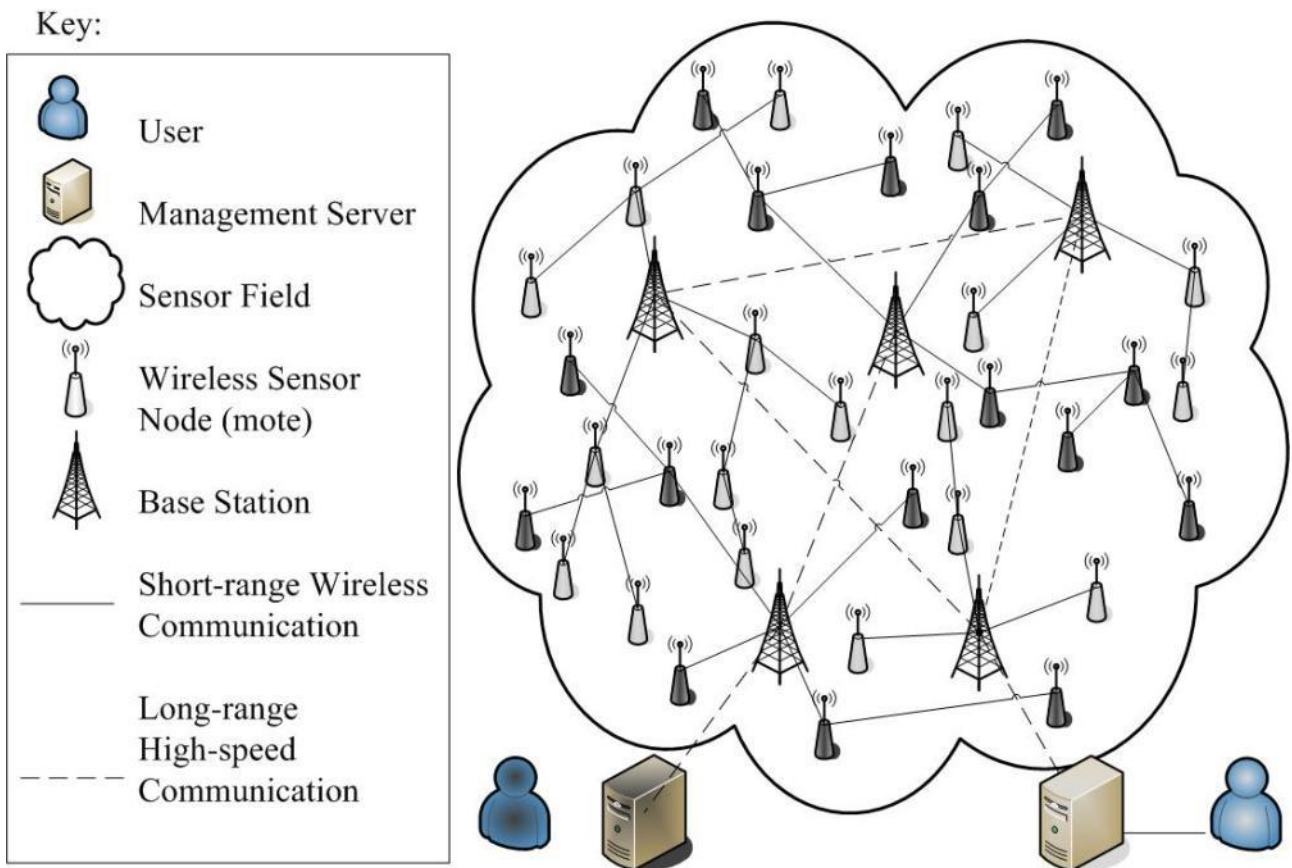


Figure 6.1. A graphical representation of an oWSN with an overlaying forensic WSN

In Figure 6.1, the darker-shaded equipment depicts the fWSN and the lighter-shaded equipment depicts the oWSN. The oWSN still communicates in the same fashion as it did previously, because the base stations and management server for the oWSN receive communication from the oWSN motes only. The darker-shaded fWSN was deployed with its own set of motes, own base stations and even its own management server. This allows us to deploy an overlaying fWSN without having to alter anything on the oWSN.

After the deployment of the fWSN motes, the fWSN will attempt a frequency-hopping technique to determine which frequency the oWSN motes are communicating on. As soon as the oWSN frequency has been determined, the fWSN will only listen for communication on that frequency and never attempt to send any data packets on the oWSN frequency. This must occur because inter-mote communication will occur on a radio frequency different from the one that is used by the oWSN. The fWSN will then attempt to sniff data packets that are broadcast from the oWSN. Once a packet has been received, the fWSN will transmit the packet in a broadcast fashion on a pre-specified radio frequency that is different from the one the oWSN uses, in an attempt to reach the fWSN base station. Different types of routing protocols can be used on the fWSN to determine the fastest path to the base station. The researcher would recommend using the security

protocol that was proposed in Mouton and Venter (2009) to transmit the packets to the base station. This security protocol was not specifically named, but it made use of the eXtended Tiny Encryption Algorithm (XTEA) which is highly efficient on WSN devices and also had the added advantage that the data packets sent between the motes are encrypted (Mouton & Venter, 2009; Hong, Hong, Ko, Chang, Lee & Lee, 2004). The security protocol proposed by Mouton and Venter (2009) also has a way of checking if the data packets can reach the base station and hence one can always be sure that every data packet received from the oWSN will in fact reach the fWSN base station and be logged. This security protocol will obviously have a negative effect on the battery life of the fWSN as the security protocol requires more processing than a protocol without these features. The negative effect can be overcome if the fWSN running this security protocol is fitted with an extended battery pack.

The form of checking referred to above is achieved by using an encryption protocol for the transmission of the data packets. Upon each data packet transmission from a mote in the WSN, the mote will first attempt to determine if it can reach the base station. The mote will transmit a HELLO packet to the base station, to which the base station must respond. The mote in the field that sent the HELLO packet will wait for the response from the base station. Only after a response has been received, will the mote send the original data packet towards the base station. By using this technique one will always be assured that the data packet is able to reach the base station.

The next section discusses the actual digital forensic readiness requirements.

6.3 Adhering to the digital forensic readiness requirements

This section shows how the proposed model adheres to the digital forensic readiness requirements that were set out in the previous chapter.

6.3.1 The broadcasted communication from the oWSN should be intercepted in a forensically sound manner.

To capture data packets in a forensically sound manner, the model is designed to cover the entire oWSN. We need to be certain that every mote in the oWSN is at least covered by a single fWSN mote. Also, the data packets captured may not be altered by the motes in the fWSN.

6.3.2 While intercepting communication, there should be no extra network overhead on the oWSN.

The model was designed in such a way that the fWSN motes will always use a different radio frequency on which to communicate. The fWSN motes are only allowed to listen in on the radio frequency used by the oWSN.

6.3.3 The fWSN should by no means be able to influence the oWSN or influence any sensory data transmitted within the oWSN.

The model is designed in such a way that all data packets captured from the oWSN are briefly saved on the fWSN mote. These packets are captured in full before they are sent onwards to the fWSN base station.

6.3.4 The fWSN should not increase the power consumption in the oWSN and the fWSN should have at least the same network lifetime or longer than the oWSN in terms of battery power.

The model as proposed earlier consists of an fWSN that is deployed either at the same time as the oWSN or afterwards to provide the added benefit of having an fWSN. In both these scenarios the fWSN would mostly be deployed on the same type of WSN equipment and thus the battery life should be relatively equal. In the proposed model, each mote in the fWSN will only communicate as often as it receives packets from the oWSN. The battery drain should consequently be equal on both the oWSN and the fWSN, and in all scenarios the fWSN will have the same lifetime or longer than the oWSN.

6.3.5 The fWSN should be able to capture all possible types of communication that can be sent from the oWSN.

The model is designed in such a way that it only relies on the ability of the equipment to capture all different types of communication. The model therefore inherently adheres to this requirement. We can also assume this to be no issue as the equipment for the oWSN and the fWSN would most likely be the same.

6.3.6 The fWSN should be able to communicate on the same radio frequencies as the ones that the oWSN is capable of using.

This is also a feature that relies on the type of equipment that is used. Thus this requirement is inherently adhered to in the model.

6.3.7 All communication within the fWSN should occur on a frequency that is not utilised in the oWSN.

The model proposes injecting an entire secondary WSN into the oWSN and thus this secondary network, referred to as the fWSN, should not influence the oWSN. To meet this requirement, the fWSN equipment must be set up to use a different radio frequency for communication than the one used by the oWSN.

6.3.8 The fWSN should be designed in such a manner that the network topology or the routing protocol used by the oWSN does not influence the fWSN's operation.

The model is proposed in such a way that it relies on the equipment to capture the data packets. Hence the equipment should be programmed in such a way that all types of communication would be captured, without taking into consideration what the communication holds. The communication that is received is never executed and thus even if malicious injections enter the fWSN, they would only be logged and not have any effect on the fWSN.

6.3.9 If an intruder WSN is in the area and communicates on a frequency that influences the oWSN, then the fWSN should be able to forensically capture these data packets.

In the model, the fWSN nodes are scattered around the oWSN nodes. Thus all communication that can be received by the oWSN nodes would also be in range of the fWSN nodes and if an attack were to occur, the fWSN would be able to log this attack. The attack can then later be verified by looking at the log files on the fWSN management server.

6.3.10 It should be possible to implement the fWSN without any modification of the oWSN.

This is the main requirement to consider while designing the model. The model is designed in such a way that the fWSN can be implemented without having to alter the oWSN. This is achieved by adding a secondary network, referred to as the fWSN, into the field.

6.3.11 The authenticity and integrity of all the data packets should remain intact while they are being captured on the fWSN.

The proposed model inherently adheres to this requirement, as it focuses more closely on how the fWSN nodes are set up. The fWSN nodes are set up to capture oWSN data packets, in full, up to and including the last bit of each data packet. Only after each successful data packet capture does the fWSN attempt to retransmit this packet onwards towards the fWSN base station.

6.3.12 The data packets that are captured in the fWSN should be stored in such a way that their authenticity and integrity are not compromised.

The storing of the data packets is performed by the fWSN management server. A physical link connects this server to the final base station. It receives all the data from the final base station and stores the full data packets, along with a hash of the data packets, in log files on the management server.

6.3.13 It should be possible to verify the authenticity and integrity of all the data packets in case a digital investigation takes place.

The full data packets are stored on the management server. Thus there are several ways to verify the authenticity and integrity of these packets. One of the ways is to match the data packets with the logs from the oWSN management server. Another one is to replay the information all through a test network and see if the same results are obtained as were obtained by the oWSN. The model proposes to verify the authenticity and integrity of packets by examining how many fWSN motes were able to capture (according to a forensically sound or proven method) a single data packet in the oWSN. If a single data packet is forensically captured by more than one fWSN, it will have more authenticity and integrity than a data packet that was transmitted only within range of one fWSN mote.

6.3.14 The data packets should have a time stamp assigned to them that does not violate their authenticity and integrity.

It is the task of the management server to assign the time stamps for each data packet that arrives. In the model this rule is adhered to as the management server is physically connected to the final base station and will thus receive all the data packets. Upon receipt of the data packets at the final base station, the management server would then save each data packet along with the time stamp generated at the time of receipt by the final base station. This procedure is adhered to because the management server is the only device in the model that has a true sense of time.

6.3.15 The sequence of the packets captured should reflect the true sequence in which they were transmitted from the original network.

The model prescribes that, after each receipt of a data packet by the fWSN, the fWSN mote has to retransmit this packet onwards towards the fWSN base station. Hence, the sequence of the packets will not be influenced by the fWSN because, whenever a data packet arrives, it will immediately be transmitted towards the fWSN base station.

Having taken all the aforementioned requirements into consideration, we can see that the model was designed in terms of the special set of requirements that an IEEE 802.15.4 WSN has in respect of digital forensic readiness. This leads us to assume that the model is indeed good enough to be used in an IEEE 802.15.4 WSN.

Now we have a model that adheres to all the requirements for a WSN to achieve digital forensic readiness, and the next section will therefore conclude this chapter.

6.4 Conclusion

Chapter 6 proposed a model that can be used to achieve digital forensic readiness in a WSN environment. This model is based on the idea that if an overlaying fWSN is used to achieve digital forensic readiness, then most of the pitfalls in WSNs can be avoided.

This chapter also used the list of factors proposed in the previous chapter to explain and show that the model will indeed work. Each factor was considered as a separate entity and then used to discuss how the model adheres to it. Having shown that all the factors can be adhered to by using a model to achieve digital forensic readiness, this dissertation will now continue to discuss in more detail how a prototype of the model was designed and implemented.

Part IV will focus exclusively on the implementation of the prototype and how it performed in demonstration scenarios.

Part IV: Prototype

Chapter 7 **Prototype Equipment**

7.1 Introduction

This chapter introduces the reader to the equipment that was used to implement the prototype. It starts off with a brief discussion of the actual equipment, followed by a discussion of the programming language and environment. Imote2 Sensor Motes and TelosB Sensor motes were used to illustrate all the scenarios for demonstration later on (Crossbow Technology Inc, 2007; Crossbow Technology Inc, 2005).

Section 7.2 is dedicated to introducing the reader to the Imote2 Sensor motes, which were the main types of WSN equipment used. Both the oWSN and the fWSN were implemented on these devices. Once the reader has been familiarised with the Imote2 Sensor motes, the dissertation proceeds to discuss the TelosB mote. The TelosB mote, which employs older technology, was simply used to show that the prototype could work with other types of WSN equipment as well.

Chapter 7 then continues with a brief discussion of the programming language that can be used on the equipment and focuses on the issues that were encountered during its use. Afterwards, the management server, which is used only as a central server for the WSN, is introduced to the reader.

This chapter is then concluded with a brief overview of the prototype equipment discussed in the different sections.

7.2 Imote2 Sensor Motes

The Crossbow Imote2 is an advanced sensor network node platform designed for demanding wireless sensor network applications that require high Central Processing Unit (CPU)/Digital Signal Processing (DSP) wireless link performance and reliability (Crossbow Technology Inc, 2007). The platform is built around Intel's XScale processor, PXA271 (Crossbow Technology Inc, 2007). It integrates an 802.15.4 radio (TI CC2420) with an on-board antenna. Each sensor node in the prototype consists of an Imote2 board as shown in Figure 7.1, an ITS400 sensor board as shown in Figure 7.2 and a battery board as shown in Figure 7.3. Figure 7.4 and Figure 7.5 depict a complete IMote2 with the sensor board, battery pack and Imote2 board connected to one another. Both these figures are included to give some perspective of the height and width of the motes. All measurements in the images are in centimetres and millimetres. The base station consists of an Imote2 board only, as it is powered by a USB port of a workstation and does not perform any sensory activities.

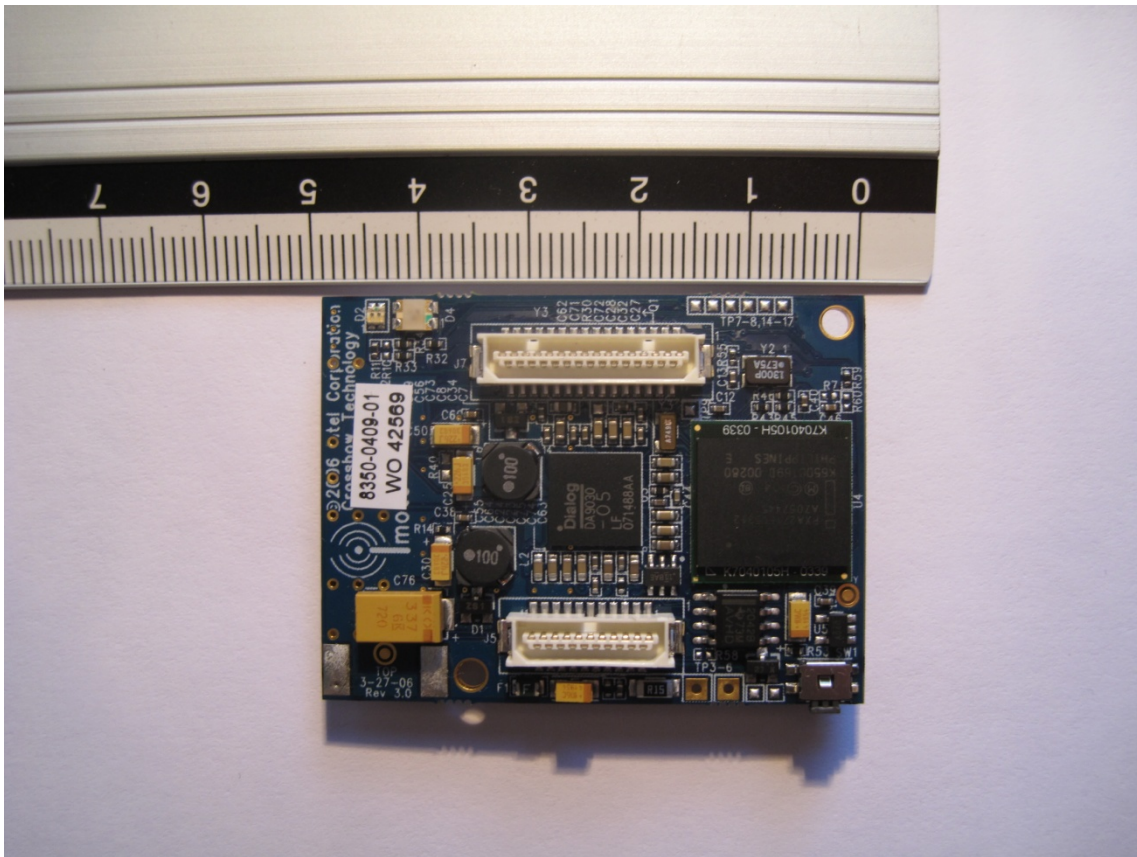


Figure 7.1. Imote2 board



Figure 7.2. ITS400 sensor board

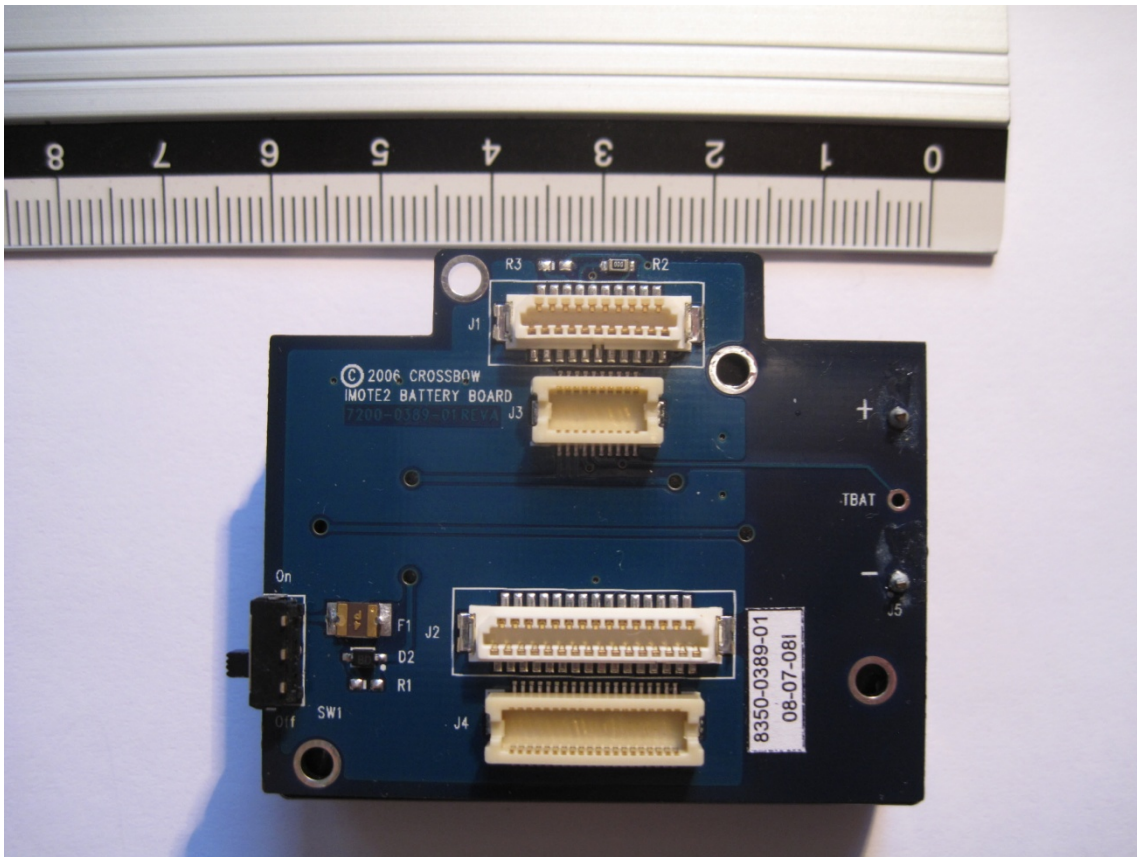


Figure 7.3. Imote2 battery board

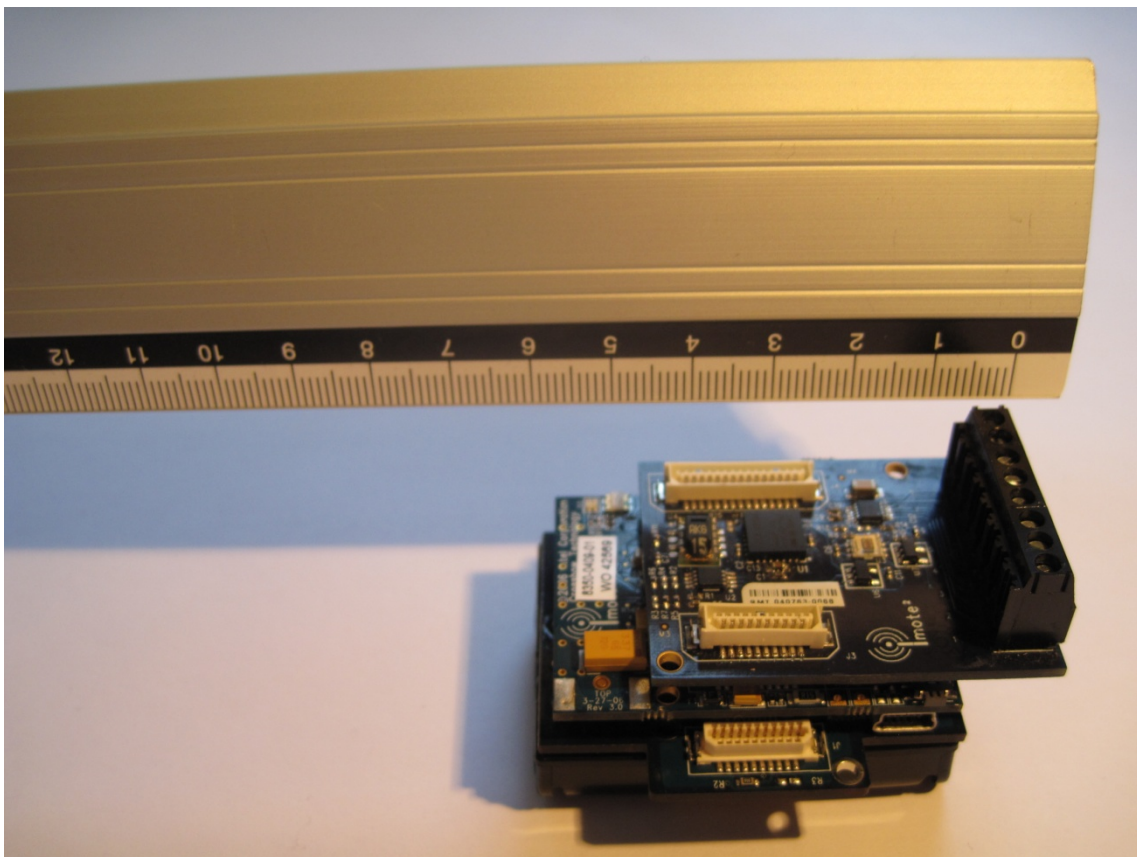


Figure 7.4. Imote2 – complete mote with all three boards plugged into one another

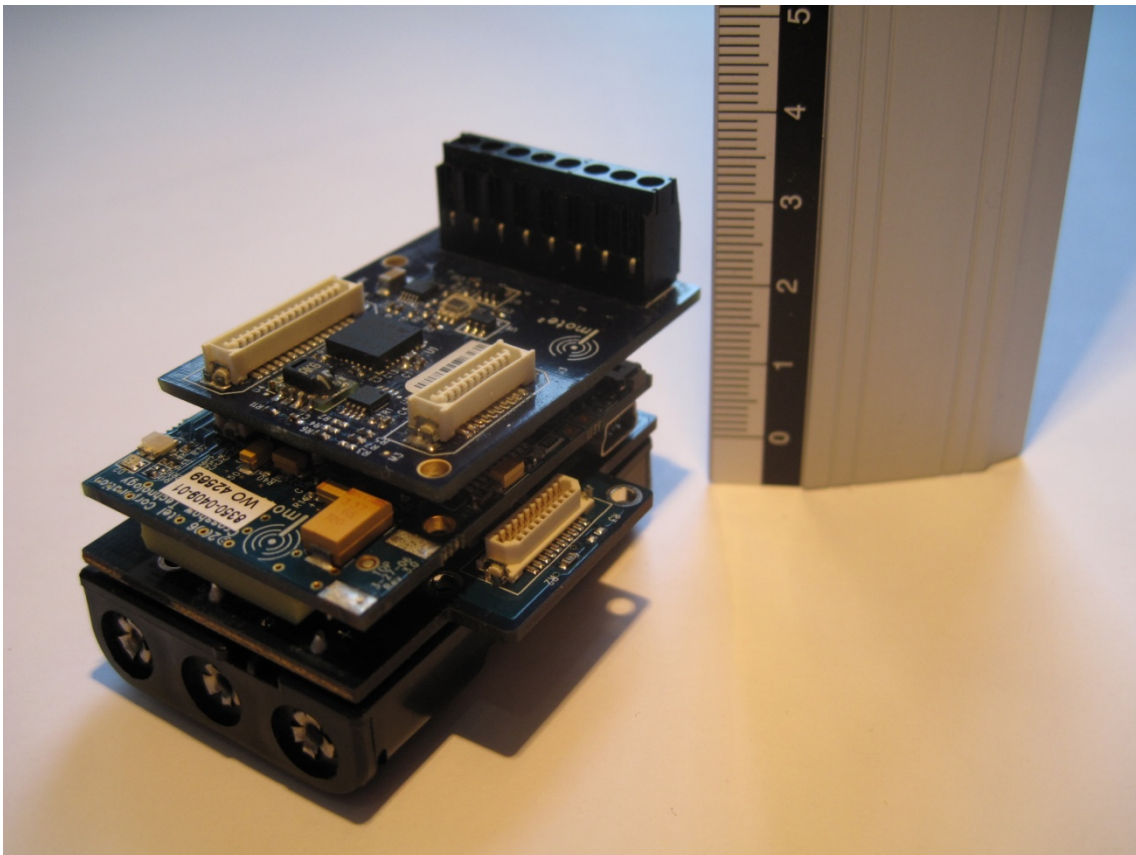


Figure 7.5. Imote2 – complete mote with perspective of height

The Imote2 mote, with its default factory settings, supports .net Micro Framework 2.0 (Crossbow Technology Inc, 2007). As these were the devices on which the prototype was implemented, all coding was done in .net Micro Framework 2.0.

The following two subsections focus on the two different boards in an Imote2 device: the Imote2 board, which does all the processing and radio transmission, and the ITS400 Sensor Board, which is only used for sensing, i.e. to measure the light, temperature and humidity around the Imote2 board.

7.2.1 Imote2 Board

The Imote2 board is a relatively advanced processing board that displays the following features (Crossbow Technology Inc, 2007):

- PXA271 XScale Processor running in a range of 13-416 MHz
- Wireless MMX coprocessor
- 256kB Static RAM, 32MB FLASH, 32MB SDRAM
- Integrated 802.15.4 radio, support for external radios through Secure Digital Input Output (SDIO) and Universal Asynchronous Receiver/Transmitter (UART)

- Integrated 2.4GHz antenna
- Multicolour status indicator LED
- Basic and advanced expansion connectors supporting 3x UART, Inter-Integrated Circuit (I2C), 2x Serial Peripheral Interface Bus (SPI), SDIO, Integrated Interchip Sound (I2S), AC97, USB host Camera I/F, General Purpose Input/Output (GPIO)
- Mini-USB port for direct PC connection
- Size: 48 mm x 36 mm. PCB Thickness 1.75 mm

Each of these motes can be assigned a MoteID for operation within the sensor field and to differentiate between different motes. The next subsection discusses the sensor board that can be attached to the Imote2 board to provide an interface for sensing data that can be measured by the sensors on the sensor board.

7.2.2 ITS400 Sensor Board

The ITS400 sensor board is shipped together with the Imote2 board in the Imote2.builder kit. The ITS400 sensor board is a multi-sensor board that combines a popular set of sensors for wireless sensor network applications, including Crossbow Technology Inc (2007) and Doolin and Sitar (2005):

- ST Micro LIS3L02DQ 3d 12 bit $\pm 2g$ accelerometer
- High Accuracy, $\pm 0.3^{\circ}C$ Sensirion SHT15 temperature/humidity sensor
- TAOS TSL2651 Light Sensor
- Maxim MAX1363 4 Channel General Purpose analog-to-digital converter for quick prototyping
- TI Tmp175 Digital Temperature Sensor with two-wire interface

Now that we have introduced the reader to the Imote2 sensor motes, we will devote the next section to the TelosB Sensor Mote.

7.3 TelosB Mote

The Crossbow TelosB Mote (TPR2420) is a much more primitive device than the Imote2 Sensor Mote. It was one of the first series of motes released by the Crossbow Company. The specifications for this mote are as follows (Moteiv Corporation, 2004):

- IEEE 802.15.4/ZigBee compliant radio frequency transceiver

- 2.4 to 2.4835 GHz, a globally compatible ISM band
- 250 kbps data rate
- Integrated on-board antenna
- 8 MHz TI MSP430 microcontroller with 10kB RAM
- Low current consumption
- 1MB external flash for data logging
- Programming and data collection via USB
- Optional sensor suite including integrated light, temperature and humidity sensor (TPR2420)
- Runs TinyOS 1.1.10 or higher

From these specifications it is clear that the TelosB Mote is inferior to the Imote2 sensor mote. The TelosB mote also does not support .net Micro Framework 2.0. The TelosB mote supports TinyOS (Moteiv Corporation, 2004; Crossbow Technology Inc, 2005), which is an operating system designed specifically for small devices that need to have limited processing and power sources.

The TelosB mote does not have extra sensor boards that may be attached. The TPR2420 model has an imbedded sensor board on the device, which can measure temperature, light and humidity. The temperature, light and humidity sensors are very similar to the ones on the Imote2 Sensor Board. Figure 7.6 shows what a TPR2420 TelosB mote looks like.



Figure 7.6. TPR2420 TelosB mote

For the purpose of this dissertation and prototype, no further information on the workings of TinyOS and the mote itself is required. Only a single TelosB mote was used throughout all the demonstration sessions. The researcher was not allowed to make any modification to the mote in terms of the software deployed on it prior to conducting this research. The researcher was also not briefed on the software on which the TelosB mote was currently running. The purpose of the TelosB mote in the demonstrations was simply to show that the prototype is able to capture data packets from oWSN devices that communicate within the same frequency range that the fWSN can listen in on.

Considering the fact that we never needed to implement anything on the TelosB mote, the following section will cover only the .net Micro Framework 2.0 as operating system and not the TinyOS as well.

7.4 .net Micro Framework 2.0

The .net Micro Framework 2.0 was already fairly dated framework at the time of writing this dissertation, and was released in September 2007 (Thompson & Miles, 2007). The .net Micro Framework 3.5 has also been released for quite some time now, but the Imote2 Sensor motes still only support up to .net Micro Framework 2.0 (Crossbow Technology Inc, 2007). There has been

talk about the Crossbow company releasing firmware to upgrade the motes to .net Micro Framework 3.0 and later, but up to the time of implementation of the prototype, this has not yet happened (Vibration Engineering Research Section, 2009).

The .net Micro Framework 2.0 is a standard development kit (SDK) to develop applications for WSN devices. The .net Micro Framework is aimed at devices that have very little processing power and storage on the devices themselves. This framework includes a very limited subset of the functionality compared to the standard .net Framework (Thai & Lam, 2003). Even though the Imote2 Sensor motes are supposed to natively support the .net Micro Framework 2.0, there are still various bugs and issues when actually deploying the code onto the devices. While the researcher was coding the prototype, he ran into several issues in this regard.

By using the fWSN, the researcher attempted to gather information about where other devices in the network were located. For this he used the received signal strength indication (RSSI) value, which is a measurement of the power level of a wireless signal received by the antenna. The RSSI value could in turn be used to determine the positioning of motes in the field by means of triangulation (Hartley & Sturm, 1997). Triangulation can be calculated if the signal strength, having been translated into the physical distance between two devices that are communicating to a third device, is known (Hartley & Sturm, 1997; Xiang-Yang et al., 2003; Chun-Hsien, Kuo-Chuan & Yeh-Ching, 2007). Merely figuring out how to access the RSSI value proved to be tricky, as the functionality was not part of the .net Micro Framework 2.0 specification. The RSSI values that were returned initially were in two's-complement notation and had to be converted to decimal values. After converting them, the researcher discovered that the RSSI values were not really true representations of the distance between the devices, as he obtained almost random values while examining them. The researcher later discovered, through the use of the Imote2 online discussion board (Vibration Engineering Research Section, 2009), that the RSSI values were not accessible using the .net Micro Framework 2.0 firmware that is currently on the motes. The Imote2 discussion board was provided by Crossbow and acted as the main forum to discuss development woes and accomplishments on the Imote2 devices. This could also be seen in the fact that the demonstration software that accompanied the device used random values when supposedly "determining" the RSSI values.

As the equipment is fairly expensive, the researcher opted to not attempt to change the firmware on the devices and went ahead without using the RSSI values. This luckily did not hamper the proof of concept for the proposed model in any way. It did, however, cause the researcher to restrict some of the features of the prototype, as triangulation could not be performed accurately on

the devices. This caused the exclusion of geographically pinpointing single motes from within the oWSN. If it was implemented on other hardware, the prototype would theoretically be able to determine the positioning of the oWSN motes. This limitation, however, does not hamper the prototype, as the purpose of the prototype was solely to demonstrate that a digital forensic readiness layer and a means of flooding detection can be added to an existing WSN.

The next section discusses the management server, which acts simply as a central point in the WSN.

7.5 Management Server

The management server to be used with WSN devices can be any computer fitted with Universal Serial Bus (USB) ports. The researcher employed his laptop computer to act as the management server.

The laptop is an HP Pavilion DV9500 series computer and has an Intel Core2Duo 1.8 GHz CPU with 2GB of RAM. The specifications of the management server are not really that important as the software written for the prototype is capable of running on most of the entry level computers one would be able to purchase today. The only important factor about the management server is that it should at all times be connected to a device that will be able to give the computer access to a forensically trusted time server. A forensically trusted time server can be defined as a server that is independently able to provide the management server with a time stamp originating from the independent time server itself. For the purpose of demonstrations, the time-a.nist.gov time server was used, as it is a widely used and accepted public time server (Mills, 1991) provided by the National Institute of Standards and Technology.

Throughout all the demonstrations the management server was connected via a USB cable to an Imote2 base station. The Imote2 base station was merely an Imote2 board with specialised software, which was also developed as part of the prototype deployed on it. Software developed as a component of the prototype was deployed on the management server to forensically capture and log all the data received by the fWSN base station.

The next section concludes this chapter with a brief overview of the equipment used.

7.6 Conclusion

This chapter was devoted to introducing the reader to the equipment that was used to implement the prototype. The use of iMote2 motes was a good decision as a platform for implementation as the coding could be done in .net Micro Framework. This allowed the researcher to easily do all the development as he was fairly familiar with Microsoft Visual Studio 2005, which did not require a

steep learning curve for the .net Micro Framework. The TelosB mote was also a good addition to the demonstrations as it could be used to show that the prototype was able to monitor data packets from different types of WSN equipment, i.e. different from the iMote2 hardware. This indicated that the prototype was also versatile and scalable as far as oWSN hardware is concerned.

In the next chapter the prototype will be discussed and demonstrated with pre-setup demonstration scenarios.

Chapter 8 **Prototype Overview**

8.1 Introduction

This entire chapter is devoted to introducing the reader to the prototype and demonstration scenarios are used to illustrate its inner workings. It starts off by providing the reader information regarding the various devices that were used for various purposes during the demonstrations. The chapter then discusses the environment in which all the tests were performed and shows that the scenarios were sufficient to demonstrate the workings of the protocol.

Section 8.2 explains the demonstration environment, the decision making within this environment and the placement of motes. Section 8.3 then discusses in three separate components how the prototype was developed and Section 8.4 provides a brief summary of the chapter.

8.2 Demonstration Environment

The demonstrations were all completed inside a research computer laboratory. This laboratory allowed the simulation of several different scenarios that are similar to real-life environments.

Due to limited funding and the costs involved in acquiring WSN devices, only two Imote2 starter kits were purchased at a cost of US\$1000 each. Each contained three battery packs, three Imote2 boards and two Imote2 sensor boards. This allowed the researcher to have two separate WSNs, each with two field motes and one base station respectively, yet severely limited the demonstration examples to fairly simple ones. However, these limitations are acceptable for the purposes of this dissertation as the proof of concept for the model could be reached.

Two different network setups were used during the demonstration examples. The first network layout is depicted in Figure 8.1.

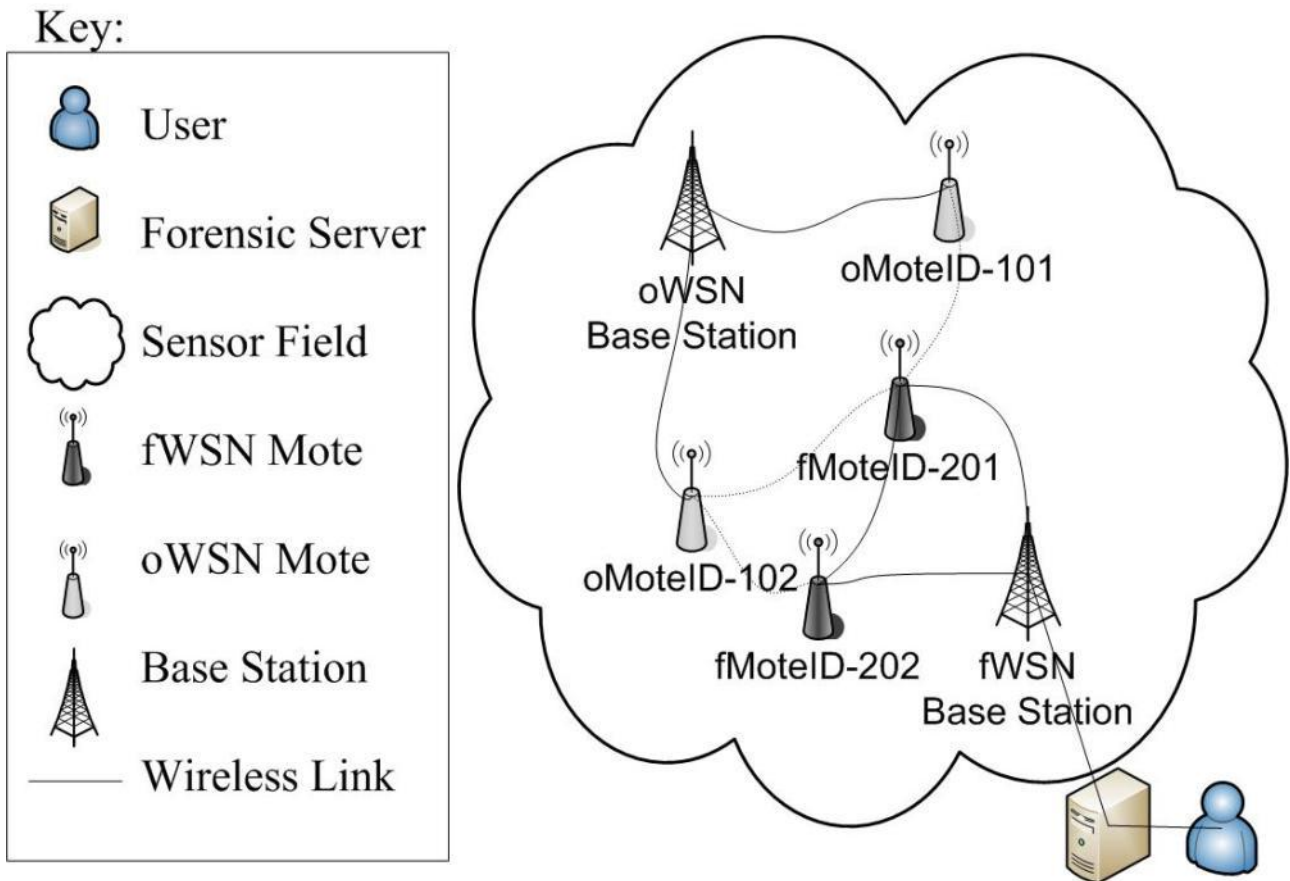


Figure 8.1. Wireless Sensor Network Layout 1

In Figure 8.1, there are two oWSN field motes and one oWSN base station, as well as two fWSN field motes and one fWSN base station. This is to simulate the oWSN co-mingled with the fWSN. The oWSN field motes are labelled oMoteID-101 and oMoteID-102 respectively, while the fWSN field motes are labelled fMoteID-201 and fMoteID-202 respectively. Figure 8.1 is set up so that broadcast transmissions from oMoteID-102 can reach only fMoteID-201, fMoteID-202 and the oWSN Base Station. Similarly, oMoteID-101's broadcast transmissions can reach only fMoteID-201 and the oWSN Base Station. The network layout is set up in this way for the demonstrations to show how the authenticity and integrity measurements are conducted and what happens when oWSN motes are on the outskirts of the fWSN.

A second network layout was used to demonstrate the normal operation of WSNs and depict a typical WSN. This layout can be seen in Figure 8.2.

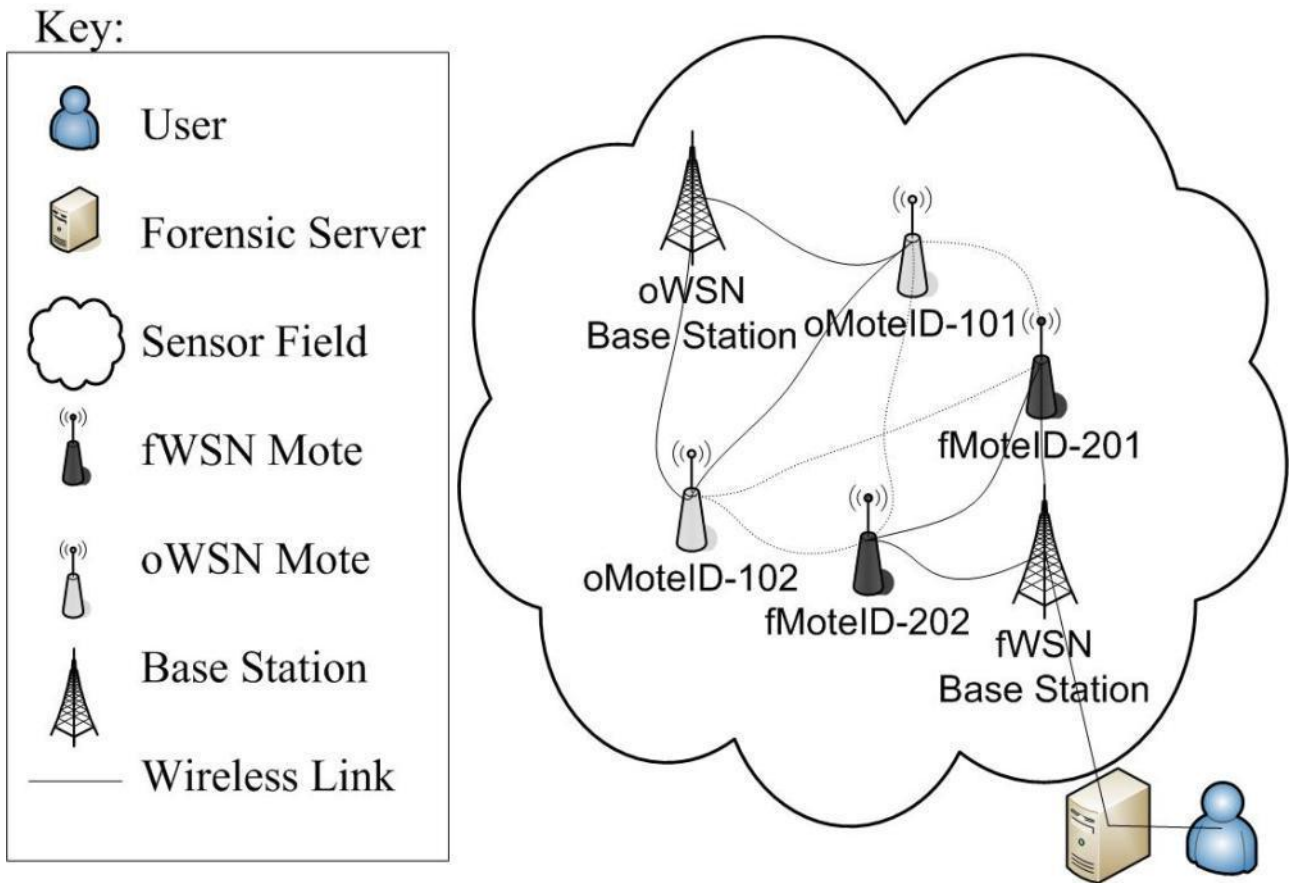


Figure 8.2. Wireless Sensor Network Layout 2

This second network layout is very similar to the first. Only a slight adjustment was made to it by placing oMoteID-101 within the range of fMoteID-202 and oMoteID-102, instead of having oMoteID-101 on the outskirts of the network. This now causes oMoteID-101's broadcasts to be seen by oMoteID-102, fMoteID-201, fMoteID-202 and the oWSN Base Station.

These two network layouts were the only ones that could be achieved with the limited number of WSN motes the researcher had to his disposal. The researcher also found that the two network layouts used were sufficient for the demonstration purposes of this research, as it satisfied the proof of concept. Throughout the demonstrations, reference will each time be made to the network layout applicable to the particular demonstration. For some demonstrations the network layout is slightly altered when one of the oWSN motes are removed. The removed oWSN mote is then reprogrammed with other software to simulate flooding.

It was important to have these different network layouts to show that the prototype is robust and could operate efficiently in several WSN environments. Using these different network layouts enabled the researcher to simulate different scenarios.

It was also important to consider the physical environment and distances that these motes were placed from each other in order to get a perspective of how a WSN operates in the two different network layouts. Figure 8.3 and Figure 8.4 are included for these purposes.

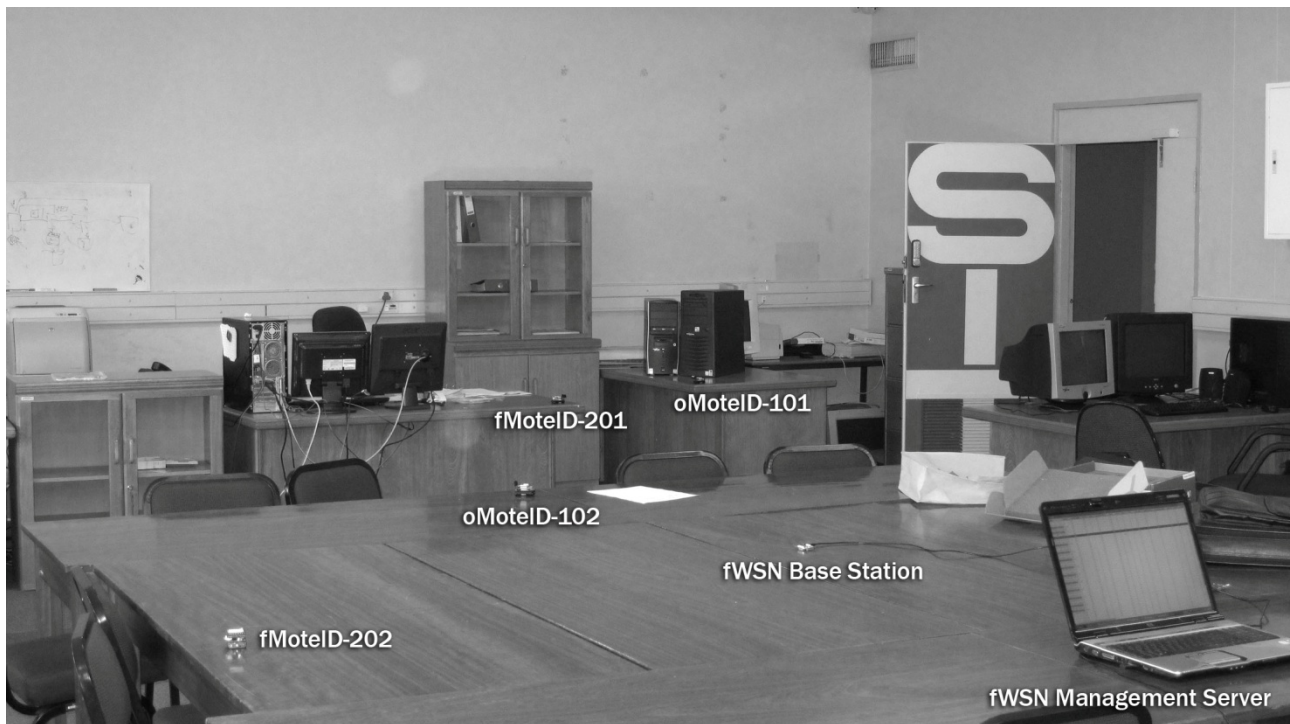


Figure 8.3. Network Layout 1 represented graphically

In Figure 8.3 we can see how the network layout in Figure 8.1 is deployed inside the research laboratory. The motes were placed far enough apart so that oMoteID-101's broadcast communication could not reach fMoteID-202. Figure 8.3 also shows how the fWSN base station is connected to the fWSN management server.



Figure 8.4. A closer look at how fMoteID-201 and oMoteID-101 are spaced apart

In Figure 8.4 we can see fMoteID-201 in front and oMoteID-101 in the rear of the image. In order to achieve the network layout depicted in Figure 8.2, one would simply swap the positions of fMoteID-201 and oMoteID-201. This would allow oMoteID-101 to be close enough to fMoteID-202 (shown in Figure 8.3), in order that fMoteID-202 can forensically capture (with authenticity and integrity as discussed earlier) the information that was broadcast from oMoteID-201. It is extremely difficult to show in the images the range of the separate motes, and a method of trial and error is therefore used in order for the researcher to place them at the correct positions, i.e. so that the motes are either in radio range of each other or not.

Now that the placement and network layout of the demonstration has been discussed, focus is shifted to how the prototype was developed.

8.3 Prototype development

The prototype was developed as three separate components. For the purpose of this study, the researcher needed to implement a prototype for each of the three components of the fWSN. The first to be developed was the software for the forensic motes in the field. The second part involved the software that was deployed onto the base station in order to retrieve the data from the forensic motes in the field. The final part was the software that was deployed on the management server, which had to log the forensic data that the base stations received from the forensic field motes.

The implementation of each of these components is separately discussed in the following subsections.

8.3.1 fWSN Field Motes

fWSN field motes required an implementation that would allow them to observe communication and capture any type of data packets. The full implementation on these motes was a fairly simple task, which required listening for packets, altering the radio frequency to ensure that it does not interfere with the oWSN motes and transmitting the packets onwards towards the fWSN base station.

Firstly, the fWSN field motes are to be deployed in such a manner that the coverage area of the fWSN field motes will always encompass the coverage area of the oWSN. Upon power up of the fWSN field motes, they would immediately switch to promiscuous listening mode. These motes would then sit in an almost idle state with the radios in listening mode. The motes were set to promiscuous listening mode to forensically capture any data packets, even if the packets were not meant for them. If promiscuous mode was not used, we could end up ignoring data packets. Thus, promiscuous listening mode was found to be the best way to capture all data packets in range.

It is not required for the fWSN field motes to negotiate a communication protocol, as the fWSN field motes rely on broadcasted messages in order to communicate with one another. The broadcasting of messages allows the fWSN field motes to avoid the possible problem which a connection-based protocol might have. In a connection-based protocol it is required for devices to continuously broadcast additional messages in order to maintain the connection. Broadcasting the messages between the fWSN field motes allows maintenance of a connectionless communication protocol which then, in turn, minimises the additional network traffic that the fWSN field motes could possibly generate.

Upon receipt of a data packet, the fWSN field mote would attempt to interpret the header of the received message. All the Crossbow WSN motes also have a software interface for the TOSRadio, which allows one direct access to the radio device on the mote board (Crossbow Technology Inc, 2005). The TOSRadio interface is the default interface used by the .net Micro Framework to interface the hardware on the Crossbow motes. Thus, data packets originating from a Crossbow WSN mote could potentially have a TOS header and a sensor board header added to it. For this reason, we analysed the packet header as it contained information about where the packet originated, the destination of the packet, the length of the packet and a 2-byte counter to display the sequence number of the packet. Data packets sent in the oWSN did not necessarily need to contain the TOS headers for the fWSN being able to capture such packets and, thus, there are

demonstrations of packets that contained either TOS headers or randomised headers. (Examples of the TOS header packets are shown later when the prototype is demonstrated.)

After an attempt was made to interpret the header of the data packet, the fWSN field mote would alter its radio frequency to a pre-specified one that is known not to clash with the radio frequency of the oWSN. This radio frequency would be any frequency that the fWSN mote was able to communicate on, while care would be taken to ensure that this frequency was not the same as the frequency used by the oWSN. The header of the data packet would next be combined with the original received data packet and added at the end of the fWSN data packet. The fWSN field mote would then use the fWSN sensor board header to set the ‘Node ID’ field into its own MoteID. This is done to analyse where the data packet originated in the oWSN. Figure 8.5 depicts the fWSN data packet, once it has been combined with the oWSN data packet.

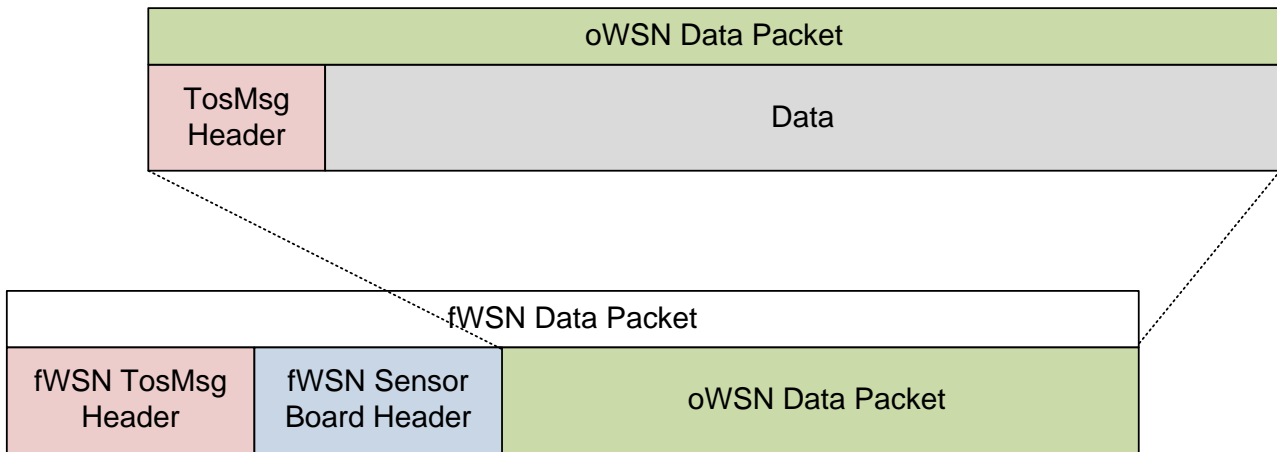


Figure 8.5. fWSN Field Mote Data Packet

The data packet would next be transmitted onwards, in a broadcast fashion, towards the fWSN base station. After this transmission, the data packet is firstly removed from the fWSN field mote and then only does the fWSN field mote return to listening for data packets in a promiscuous mode. This deletion of the data packet after the transmission allows one to avoid exceeding the maximum data buffer of an fWSN field mote.

The next subsection is devoted to exploring how the fWSN base station was implemented.

8.3.2 fWSN Base Station

The fWSN base station was designed in such a way that it would not be allowed to transmit any data packets, since its purpose was to simply collect the forwarded packets from the fWSN notes. The fWSN base station is always set to listening mode on the specific fWSN radio frequency.

The fWSN base station operates in a simple fashion in the sense that it only listens for data packets on a specific predefined radio frequency. This approach causes no interference with the oWSN. The fWSN base station would also be unable to receive any communication from the oWSN as it is not set to listen in promiscuous mode; neither is it set to listen on the oWSN radio frequency (as mentioned before).

The fWSN base station will, upon receipt of a data packet, immediately transmit it to the management server through the physical USB connection. The physical USB connection should ensure that the transmission is always successful. In the case where the management server is down, because of a power outage or something similar, the fWSN base station will build up a buffer of data packets that have been received. In a scenario where the ram buffer is full, the ram buffer will be dumped to the flash memory of the mote and have to be retrieved manually at a later stage. As the buffer is 32 megabytes in size, it will only become full in extreme circumstances where the management server is off for several days. At the time that the management server goes online again, it will receive in one transmission all the data packets that have been captured while it was offline.

The next subsection explains the workings of the software on the management server.

8.3.3 fWSN Management Server

The management server had to have a specific set of software designed for it. This software had to be able to communicate in real-time with the base station and have a graphic display of what was happening on the network. In order to prove that the data packets were captured in a forensic fashion, the management server software was split into two separate pieces of software. The software of the management server was implemented in C# as it was the most efficient language to interface with the .net Micro Framework.

The two components of the management server software include the packet logging software and the packet analysis software. The main task of the packet logging software was simply to log all the communication from the fWSN in a forensically sound manner. The term forensically sound is used as defined in Chapter 6.3.1. The purpose of the packet analysis software was to provide a GUI for an fWSN management server administrator to easily interpret the data.

In the following two subsections, the packet logging software and the packet analysis software are explained in more detail.

8.3.3.1 Packet Logging Software

The software is programmed to open a unidirectional connection to the fWSN base station. This means that the base station is able to transmit data on this connection towards the management server, but the management server cannot communicate towards the fWSN base station. Implementation was done in this way so that if malicious software were to compromise the management server, it would not be able to spread to the fWSN base station and potentially cause further harm to the fWSN.

Upon receipt of a data packet from the fWSN base station, the management server software generated a time stamp by using the system time at that point. It is important to note that the management server's system time had to be synchronised upon initiation of the software with the forensically approved time server as explained in Chapter 5.2.3. The data packet was then stored along with the full time stamp.

These data packets are only a matter of bytes and are stored directly onto the hard disk of the management server, which is an average server with ample storage. The amount of storage on the management server thus nullifies a potential problem where the logging machine may have insufficient storage. As mentioned earlier, WSN devices are devices with low processing abilities and thus the amount of data that they possibly can transmit is extremely small. A sample of the log file can be seen in Figure 8.6.

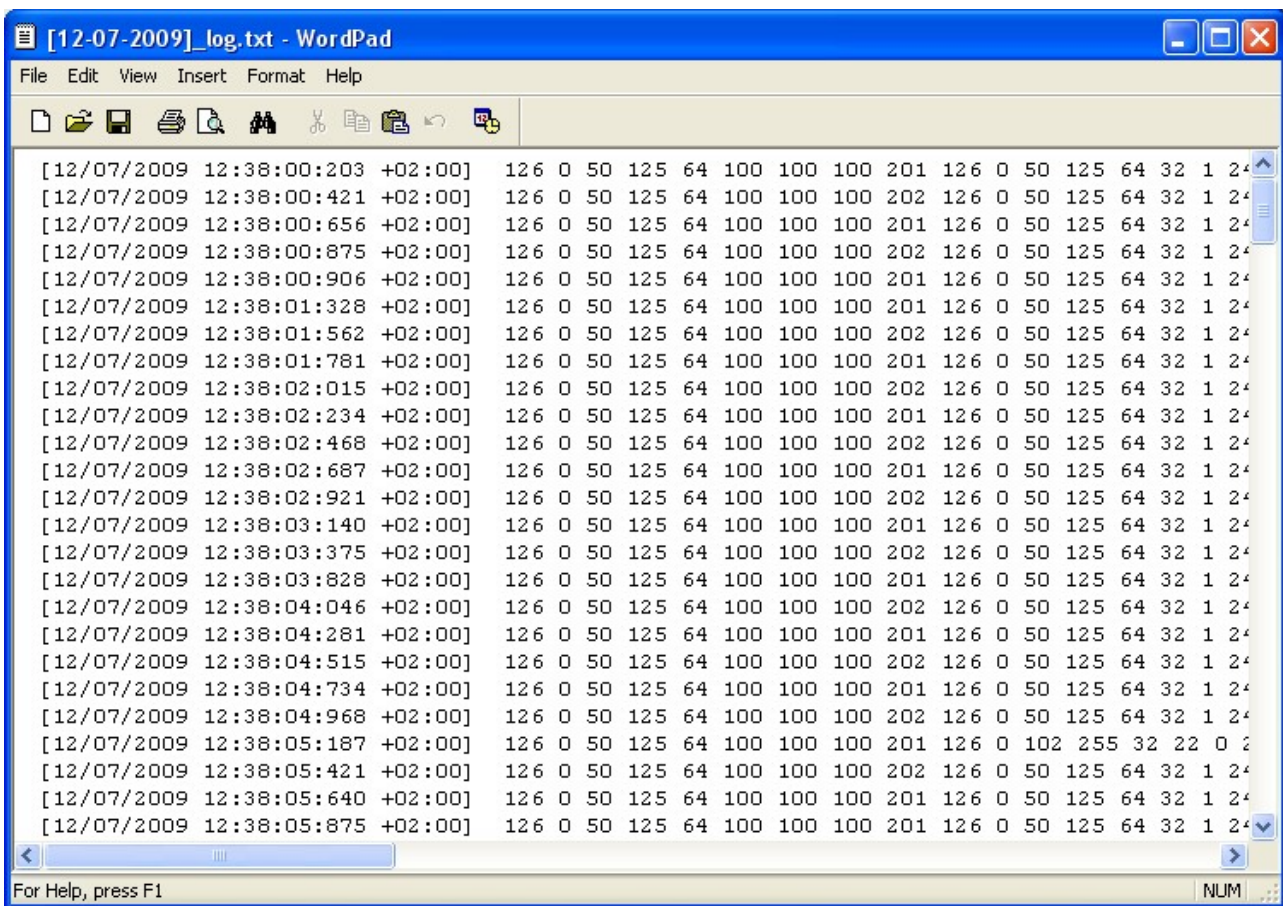


Figure 8.6. Sample log file on 12/07/2009 at 12:38:00 until 12:38:06

From Figure 8.6 it is clear that all data packets received were saved along with their full time stamps. (Later in the dissertation we will discuss how to interpret this log file.) Also, from the above, we can see that each day has its own log file as the date of capture was saved as the log file name in order to decrease the complexity of a potential digital forensic investigation.

The sole purpose of the packet logging software is to log all the data packets received by the fWSN base stations to a log file. In order to ensure authenticity and integrity, only the packet logging software has the ability to create this log file. In the next subsection, our focus is on the packet analysis software.

8.3.3.2 Packet Analysis Software

In order to retain the authenticity and integrity of the log files, the packet analysis software is implemented as a separate part that cannot alter these log files. This software is only allowed to open the log files in a read-only fashion and thus cannot alter them.

The software only serves as a graphical user interface (GUI) to the log files. It is used as a way to easily interpret the data that has been captured forensically by the packet logging software. This piece of software also provides us (in terms of digital forensic readiness) with a quicker way to

interpret the data than to manually examine the log files, which could be a tedious task. If the packet analysis software is used in a real-time fashion, one has the ability to view the data packets as they are being captured and thus see the full flow of data packets in the WSN. Apart from showing how the data packets are captured in real time, the packet analysis software also has several other features. It has built-in algorithms to detect flooding inside the oWSN. The software also filters out data packets that have been seen by more than one fWSN field mote, in order to make the information seen in the packet analysis software more easily readable. In the case where this data is filtered out, the forensic MoteIDs are displayed in sequential order to show which forensic motes were able to capture the data packet first. The closer an fWSN mote is to an oWSN mote, the faster the data packet will be captured by the fWSN mote and thus one can determine which fWSN mote was closer to the oWSN mote that transmitted the data packet. A sample of the GUI can be seen in Figure 8.7.

Line	Date	Seen By	Forensic MoteID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
3	12/07/2009 13:02:22:484	201, 202	201	126	0	47	125	31	18	0	193	27	158
5	12/07/2009 13:02:42:328	201, 202	201	126	0	47	125	31	18	0	192	75	10
7	12/07/2009 13:03:00:500	201, 202	201	126	0	47	125	31	18	0	192	22	241
11	12/07/2009 13:03:29:046	201, 202	201	126	0	47	125	31	18	0	192	11	107
13	12/07/2009 13:03:47:046	201, 202	201	126	0	47	125	31	18	0	192	86	21
16	12/07/2009 13:04:11:453	201, 202	201	126	0	47	125	31	18	0	192	65	42
18	12/07/2009 13:04:29:984	201, 202	201	126	0	47	125	31	18	0	192	207	139
22	12/07/2009 13:06:09:437	201, 202	201	126	0	47	125	31	18	0	192	50	156
23	12/07/2009 13:06:37:000	201, 202	201	126	0	47	125	31	18	0	193	102	172
1	12/07/2009 13:02:07:718	201, 202	201	126	0	102	255	32	1	0	54	32	22
2	12/07/2009 13:02:21:390	201, 202	201	126	0	102	255	32	2	0	59	28	22
4	12/07/2009 13:02:34:843	201, 202	201	126	0	102	255	32	3	0	20	27	22
6	12/07/2009 13:02:48:453	201, 202	201	126	0	102	255	32	4	0	243	27	22
8	12/07/2009 13:03:02:000	201, 202	201	126	0	102	255	32	5	0	255	27	22
9	12/07/2009 13:03:15:359	201, 202	201	126	0	102	255	32	6	0	237	27	22
10	12/07/2009 13:03:28:796	201, 202	201	126	0	102	255	32	7	0	56	27	22
12	12/07/2009 13:03:42:406	201, 202	201	126	0	102	255	32	8	0	50	27	22
14	12/07/2009 13:03:55:890	201, 202	201	126	0	102	255	32	9	0	54	26	22
15	12/07/2009 13:04:09:312	201, 202	201	126	0	102	255	32	10	0	54	26	22
17	12/07/2009 13:04:22:890	201, 202	201	126	0	102	255	32	11	0	49	26	22
19	12/07/2009 13:04:36:437	201, 202	201	126	0	102	255	32	12	0	52	26	22
20	12/07/2009 13:04:49:859	201, 202	201	126	0	102	255	32	13	0	51	26	22
21	12/07/2009 13:05:03:328	201, 202	201	126	0	102	255	32	14	0	49	26	23

Figure 8.7. Sample GUI for a log file generated on 12/07/2009 at 13:02:07 until 13:06:38

In the GUI of the packet analysis software, the “Line” field is used to indicate the sequence in which the packets were received. In Figure 8.7 the numbers are not sequential, as the data was sorted by Byte 3 (see explanation below). The “Date” field is read from the log file and displayed alongside each data packet. The “Seen By” field is used to show which fWSN field motes were able to forensically capture the data packet. It is assumed that the data packet was forensically captured because of the fact that while the packet was travelling from the fWSN field mote to the fWSN base station onto the management server, it was always treated in a forensically sound manner. Thus the authenticity and integrity of the captured data packet is maintained throughout the entire process. In the example above, one can see two fWSN field motes with Forensic MoteIDs 201 and 202

respectively. The “Forensic MoteID” field displays which fWSN field mote was the closest to the oWSN mote that transmitted the data. The calculation is made on the basis of which fWSN field mote received the transmitted data packet first. The fWSN field mote closest to the transmitting oWSN mote would be able to receive the packet a few milliseconds before an fWSN field mote further away. This allows us to easily determine the vicinity from where the data packet was transmitted.

The “Byte 1” to “Byte 10” fields are the actual content of the bytes of the data packets that have been received. “Byte 1” to “Byte 5” contains the TOS header of the data packet. Due to the limited space of the page, not all of the Byte fields are displayed. The Byte fields go up to “Byte 64”, which was the maximum packet size one was able to transmit through the oWSN as it was the limitation on the oWSN motes.

Not all demonstrations will include the Byte fields, as the fields were given appropriate names in order to increase the readability of this dissertation. However, in a digital forensic investigation the fields would stay numbered from “Byte 1” to “Byte 64” as an investigator will need to examine each field. Numbering the fields will not hamper the digital forensic investigation, as it is only used to easily identify which byte segment the data packet is being shown.

As mentioned earlier, the packet analysis software is also able to detect flooding inside the oWSN. This will be explained in more detail while demonstrating that scenario in the next chapter, as flooding is a very important issue that this dissertation addresses.

The next section concludes the chapter by providing a brief overview of the prototype.

8.4 Conclusion

Chapter 8 was devoted to introducing the reader to the prototype setup. It discussed the different demonstration environments and network layouts that the prototype would be tested on. The development and setup of the prototype were also discussed in detail. The prototype was separated into three distinguishable components namely the fWSN field motes, the fWSN base station and the fWSN management server. The fWSN field motes are the motes deployed among the oWSN field motes, i.e. in the same sensor field. The fWSN base station is the mote to which all the fWSN field motes communicate and it is connected to the fWSN management server. The fWSN management server is used to forensically log the oWSN data packets and can also be used to analyse the oWSN data packets.

Chapter 9 gives an overview of the demonstrations.

Chapter 9 **Prototype Setup**

9.1 Introduction

The prototype proposed in this dissertation has a number of benefits for wireless sensor networks. The subsections that follow discuss four demonstrations that have been completed in order to focus on various features of the prototype. These demonstrations show:

- how the fWSN operates under general conditions;
- how the fWSN deals with flooding attacks on the oWSN;
- how the fWSN can be used to verify sensory data captured by the oWSN; and
- that the fWSN can be used to capture data packets from other types of WSN equipment.

To summarise, the first demonstration shows how the prototype performs under general conditions that resemble those under which most current WSNs operate. In the second demonstration this dissertation focuses on flooding in WSNs and how the prototype can be used to detect flooding in WSNs. The third demonstration is used to show that the authenticity and integrity of the data packets are retained whilst the prototype is being used. The fourth and final demonstration shows that the prototype can capture data packets from any type of WSN device as long as this device communicates within a frequency range that the prototype can listen in on.

9.2 Demonstration I: General Conditions

This demonstration shows how the prototype fares in an environment that resembles as closely as possible an environment found in most current WSN applications. For this demonstration the environment was set up as can be seen in Figure 9.1.

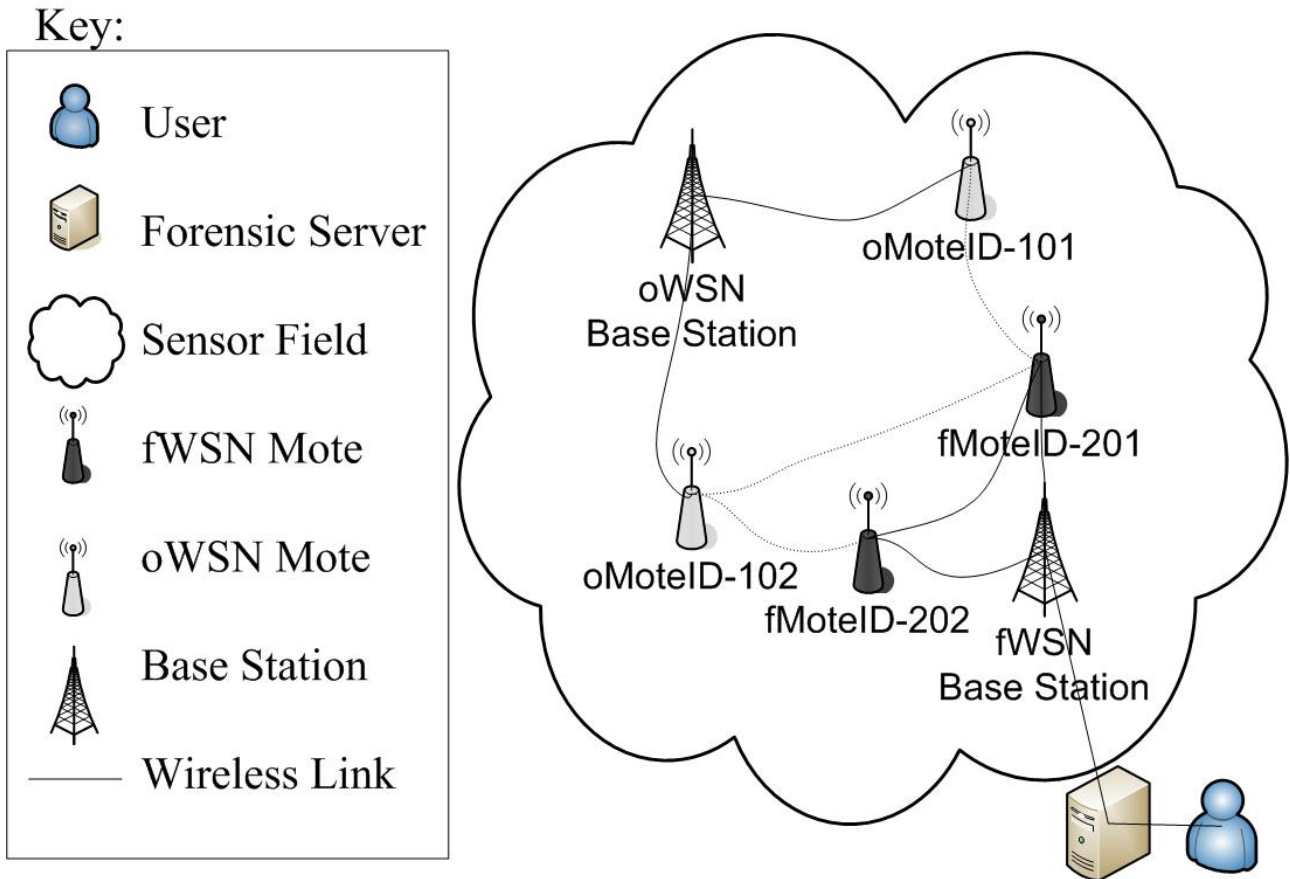


Figure 9.1. Demonstration I: Wireless Sensor Network Layout for General Conditions

WSN Layout I in Figure 9.1 is repeated here for convenience sake and in order to improve the readability of this dissertation. The oWSN consisted of two motes labelled oMoteID-101 and oMoteID-102 respectively. There was also an oWSN base station that was used to capture the data packets from the two oWSN motes. These oWSN motes were all switched on before the fWSN was deployed in order to simulate an already existing WSN. The fWSN also consisted of an fWSN base station and two motes labelled fMoteID-201 and fMoteID-202 respectively. The fWSN was deployed in a specific way in order to demonstrate two different scenarios that could occur in this setup.

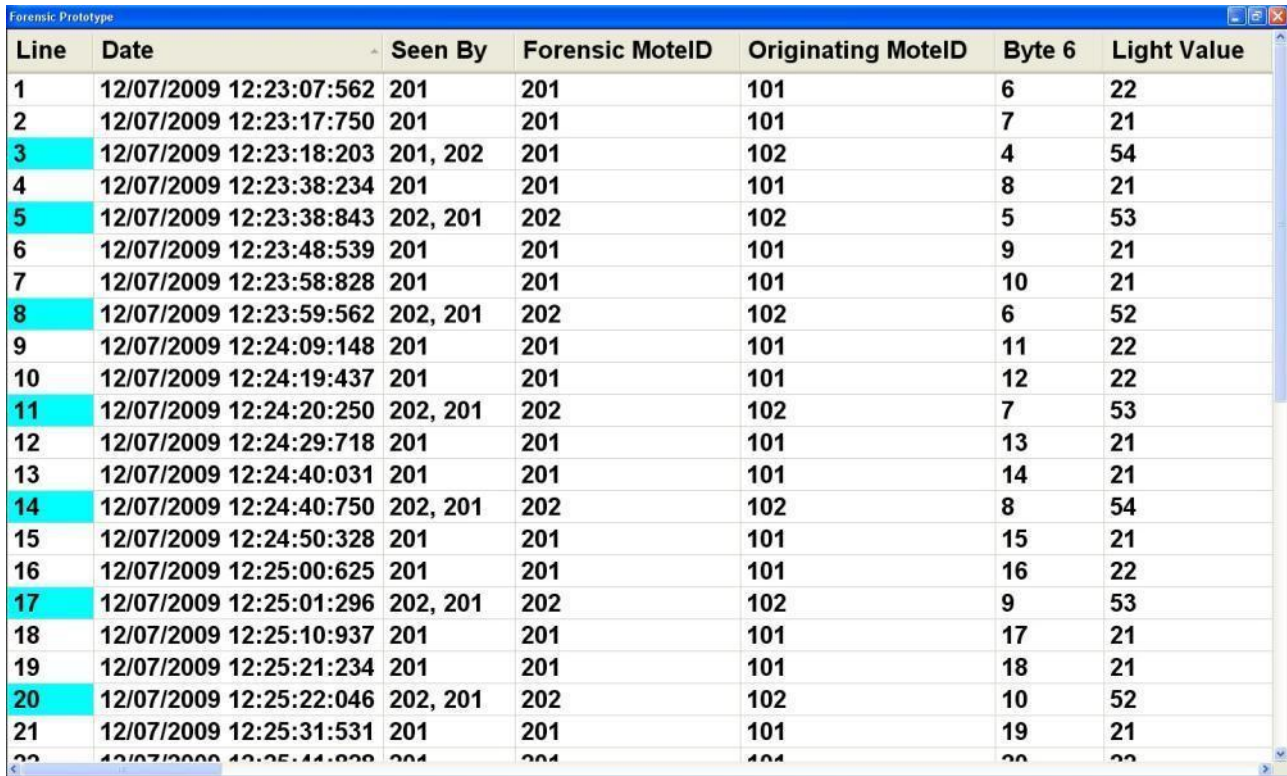
The first scenario was where an fWSN mote, fMoteID-201, was in range of more than one oWSN mote. In the second scenario an fWSN mote, fMoteID-202, was in range of only a single oWSN mote. In order to achieve this, fMoteID-201 was deployed to be in range of both oMoteID-101 and oMoteID-102, whereas fMoteID-202 was deployed to be only in range of oMoteID-102. Furthermore, fMoteID-202 was deployed far enough from oMoteID-101 so as not to be in oMoteID-101's wireless range, and close enough to oMoteID-102 to be within oMoteID-102's wireless range. The fWSN base station was subsequently deployed to be in range of both fWSN

notes and connected directly to the fWSN management server. The environment as set up graphically can be seen in Figure 9.2.



Figure 9.2. Demonstration I: Wireless Sensor Network Layout for General Conditions (Graphic Representation)

After the fWSN was fully deployed into the sensor field, the fWSN was switched on. Both the packet analysis software and packet logging software were started on the management server. A screenshot of the packet analysis software is shown in Figure 9.3.



Line	Date	Seen By	Forensic MoteID	Originating MoteID	Byte 6	Light Value
1	12/07/2009 12:23:07:562	201	201	101	6	22
2	12/07/2009 12:23:17:750	201	201	101	7	21
3	12/07/2009 12:23:18:203	201, 202	201	102	4	54
4	12/07/2009 12:23:38:234	201	201	101	8	21
5	12/07/2009 12:23:38:843	202, 201	202	102	5	53
6	12/07/2009 12:23:48:539	201	201	101	9	21
7	12/07/2009 12:23:58:828	201	201	101	10	21
8	12/07/2009 12:23:59:562	202, 201	202	102	6	52
9	12/07/2009 12:24:09:148	201	201	101	11	22
10	12/07/2009 12:24:19:437	201	201	101	12	22
11	12/07/2009 12:24:20:250	202, 201	202	102	7	53
12	12/07/2009 12:24:29:718	201	201	101	13	21
13	12/07/2009 12:24:40:031	201	201	101	14	21
14	12/07/2009 12:24:40:750	202, 201	202	102	8	54
15	12/07/2009 12:24:50:328	201	201	101	15	21
16	12/07/2009 12:25:00:625	201	201	101	16	22
17	12/07/2009 12:25:01:296	202, 201	202	102	9	53
18	12/07/2009 12:25:10:937	201	201	101	17	21
19	12/07/2009 12:25:21:234	201	201	101	18	21
20	12/07/2009 12:25:22:046	202, 201	202	102	10	52
21	12/07/2009 12:25:31:531	201	201	101	19	21
22	12/07/2009 12:25:44:000	201	201	101	20	22

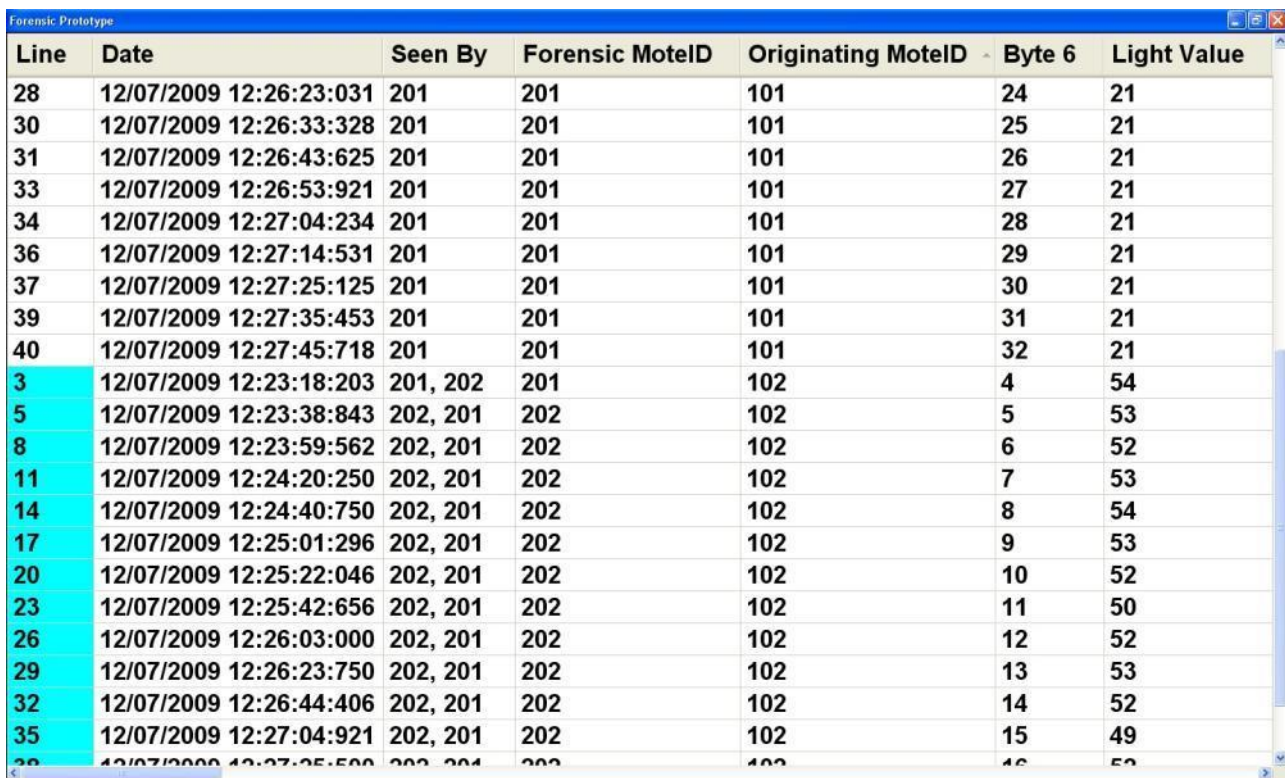
Figure 9.3. Demonstration I: Packet analysis software screenshot to demonstrate how the prototype performs under general WSN conditions

In Figure 9.3 we can see how the fWSN logged the data packets that originated in the oWSN. The “Line” column shows in what sequence the packets were received by the fWSN base station, while the “Date” column shows the exact time that the packet was captured by the fWSN. The “Seen By” column shows which fWSN motes in the field were in range and able to capture and rebroadcast the data packet so that the packets are eventually “forwarded” to the base station. In this demonstration 201 represents the ID of the mote labelled fMoteID-201 and 202 represents the ID of the mote labelled fMoteID-202 as shown in Figure 9.3.

The “Forensic MoteID” column shows which fWSN mote was the quickest to capture a specific data packet and to send it onwards towards the fWSN base station. It would appear that the “Forensic MoteID” was random at the start (see Figure 9.3), but it is important to note that two oWSN motes were being monitored. On the third line, fMoteID-201 captured the packet first and from the fifth line onwards (for oMoteID-102), fMoteID-202 captured the packet first. This was due to the fact that while running the prototype there was human movement between the motes, which caused the paths between the motes to be obstructed and thus the delay between receiving the packets to be different. This scenario was demonstrated particularly because such obstruction could very well occur in a real-life situation. It also shows an advantage of the prototype, namely that it is indeed able to detect when obstructions occurred within the communication ranges.

The “Originating MoteID” column shows from which oWSN mote the data packet was received. In this demonstration 101 represents oMoteID-101 and 102 represents oMoteID-102 as shown in Figure 9.3. The “Byte 6” column shows the sequence number of the data packet generated by the particular oWSN, as the title “Sequence Number of Data Packet” was too long to fit in the figure. It is important to note that the sequence numbers that were captured by the fWSN and displayed in Figure 9.3 do not start at 1, because the fWSN was only switched on after the oWSN had been switched on. This means that the fWSN was never able to capture the very first few packets broadcast by the particular oWSN. It was performed in this way to demonstrate that the fWSN can be deployed while the oWSN is already in operation. The “Light Value” column shows the sensor reading of the light sensor on the oWSN motes. It is important to note that there are various other fields that also form part of packet analysis software (i.e. the temperature sensor reading, humidity sensor reading, etc.). In order to retain readability of the image, however, these fields were removed from the screenshot. In a forensic analysis, however, one should consider all the byte fields (the entire table).

Figure 9.4 is provided to help explain the first demonstration.



Line	Date	Seen By	Forensic MoteID	Originating MoteID	Byte 6	Light Value
28	12/07/2009 12:26:23:031	201	201	101	24	21
30	12/07/2009 12:26:33:328	201	201	101	25	21
31	12/07/2009 12:26:43:625	201	201	101	26	21
33	12/07/2009 12:26:53:921	201	201	101	27	21
34	12/07/2009 12:27:04:234	201	201	101	28	21
36	12/07/2009 12:27:14:531	201	201	101	29	21
37	12/07/2009 12:27:25:125	201	201	101	30	21
39	12/07/2009 12:27:35:453	201	201	101	31	21
40	12/07/2009 12:27:45:718	201	201	101	32	21
3	12/07/2009 12:23:18:203	201, 202	201	102	4	54
5	12/07/2009 12:23:38:843	202, 201	202	102	5	53
8	12/07/2009 12:23:59:562	202, 201	202	102	6	52
11	12/07/2009 12:24:20:250	202, 201	202	102	7	53
14	12/07/2009 12:24:40:750	202, 201	202	102	8	54
17	12/07/2009 12:25:01:296	202, 201	202	102	9	53
20	12/07/2009 12:25:22:046	202, 201	202	102	10	52
23	12/07/2009 12:25:42:656	202, 201	202	102	11	50
26	12/07/2009 12:26:03:000	202, 201	202	102	12	52
29	12/07/2009 12:26:23:750	202, 201	202	102	13	53
32	12/07/2009 12:26:44:406	202, 201	202	102	14	52
35	12/07/2009 12:27:04:921	202, 201	202	102	15	49

Figure 9.4. Demonstration I: Packet analysis software screenshot with data sorted by the “Originating MoteID” column

In Figure 9.4 the captured data is sorted according to the “Originating MoteID” field. This demonstration shows that the prototype was able to successfully capture data packets from the oWSN.

The motivation for sorting the data in Figure 9.4 according to the “Originating MoteID” field is that the present research will first examine the traffic generated by oMoteID-101. Firstly, by examining the “date” column, one can see that oMoteID-101 sent out a data packet roughly every ten seconds. These data packets were only in range of fMoteID-201, which demonstrates a scenario where an oWSN mote is on the outskirts of the fWSN. Due to the fact that only one fWSN mote was able to capture data packets from the oWSN mote, it is difficult to verify the integrity and authenticity of the data packet (it is not possible to examine how many other fWSN motes also captured the same packet as there was only one in this case). This would require one to use another measurement to verify the authenticity and integrity of this data packet.

The authenticity and integrity of the data packet for this prototype can be determined by examining if all the data packets received from this single oWSN mote exhibit the same typical qualities and patterns as a single data packet. The qualities and patterns of the data packet from the oWSN will be unique to each layout of the oWSN, as each layout would have its own qualities and patterns. In the implementation of the oWSN considered in this dissertation, there was a sequence number field from the oWSN and a light sensor reading on the oWSN (see Figure 9.4). Thus one can say that the data packets captured retained their authenticity and integrity based on three factors.

- Firstly, the different data packets were received about ten seconds apart and this time interval was constant throughout the entire log. This means that this oWSN mote broadcasts sensory data every 10 seconds. The time intervals at which WSNs communicate would in most scenarios stay constant as variables are measured at set time intervals. It would not make sense to measure the temperature or light at random, as this would hamper the ability to draw statistics from the measured data. In the unlikely case that the time intervals are not periodic, one would rely on other sets of patterns, for example sequences of data packets or sensor measurements, in order to verify authenticity and integrity.
- Secondly, the sequence numbers of the data packets, which can be seen in the “Byte 6” column, followed a chronological order. The sequence numbers allowed the investigator to determine that the data was sent from the same oWSN mote each time, as each oWSN keeps its own sequence counter. One could assume that most oWSN implementations use

sequence numbers as this would allow the oWSN itself to keep track of the sequence of data packets.

- Thirdly, the light sensor measurement remained constant at either 21 or 22 as can be seen in Figure 9.3 and Figure 9.4. However, this measurement must also be compared to the logs of the oWSN management server so as to determine if the light sensor measurements did indeed stay constant. This examining of the logs of the oWSN management server would not encroach on the oWSN as it would only be done in the case of a digital forensic investigation, i.e., after it was determined that an investigation needs to be launched due to a suspected security breach in the oWSN.

It is important to note that the three factors above are only taken into account to verify the authenticity and integrity of the data packets during a digital forensic investigation and one would not normally, during the operation of the fWSN, need to verify this. It is only in the aftermath of a security breach that one would need to prove that the data has been forensically captured and thus would need to have access to the oWSN data.

Based on the factors above we can safely say that these data packets truly originated from an oWSN mote with a MoteID 101. In a different oWSN environment one could use other qualities or patterns of the oWSN to determine the integrity and authenticity of the data packets that have been captured. The factors used in this demonstration were factors that are applicable to this demonstration environment only. Some of these factors might repeat in other environments while additional ones could even be found in other environments. Some of these additional factors could include determining whether the humidity measurements stayed constant and equal to the ones that were measured in the oWSN, or even whether the length of the data packets stayed constant throughout the entire fWSN capture. Taking all these factors into account, one could make the claim that the data packets were captured in a forensically sound manner because the data within reflects the true data transmitted and, while capturing these data packets, the oWSN was not influenced in any manner.

We will now examine the data packets that originated from oMoteID-102 as shown in Figure 9.3. This oWSN mote was in range of both fMoteID-201 and fMoteID-202. For demonstration purposes the “Line” column was shaded grey when more than one fWSN mote forensically captured the data packet. The scenario demonstrated will be the same for all the cases where two or more fWSN motes forensically captured a data packet. One can see from Figure 9.4 that both fMoteID-201 and fMoteID-202 were able to capture the data packets that originated from oMoteID-102. Line 3 contains an example where fMoteID-201 was the fastest at sending the data packet

onwards to the fWSN base station. In all the other lines, fMoteID-202 was faster at capturing the data packet and sending it towards the fWSN base station. This phenomenon was briefly discussed while the reader was introduced to the demonstration figures in Chapter 9. This phenomenon occurs because the distance between motes affects the time it requires for the wireless signal to travel. Thus, a mote that is further away from the sender will take longer to receive the message than a mote that is closer to the sender. Due to this phenomenon, one can conclude that fMoteID-202 was indeed closer to oMoteID-102. This can be confirmed by examining Figure 9.2. The distance that fMoteID-202 and fMoteID-201 were from oMoteID-102 was, however, almost identical and that is why, when human movement between the motes occurred, there was one case where fMoteID-201 was indeed faster at transmitting the data packet to the fWSN base station. The “Byte 6” column, which contains the sequence number from the oWSN, follows chronologically and thus confirms that all the data packets transmitted were captured. As mentioned earlier while examining oMoteID-101’s traffic, it will depend on the oWSN implementation’s qualities and patterns to determine the authenticity and integrity of the data packets captured, as oMoteID-101 was on the outskirts of the network. In the case of oMoteID-102, though, we had the advantage that it was in range of more than one fWSN mote and thus one would be able to determine the authenticity and integrity by using Casey’s Certainty Scale.

At this stage it can be concluded that the prototype was successful in the demonstration because of the following factors:

- The fWSN was able to forensically capture the data packets from the oWSN, without manipulating the data packets of the oWSN.
- The fWSN captured the data packets in such a manner that the authenticity and integrity of the data packets were not hampered.
- The fWSN was deployed after the oWSN was already in operation and hence the digital forensic readiness layer can be added to an existing WSN at any time.
- The fWSN had no previous knowledge of the communication protocols used by the oWSN.
- The fWSN was able to capture data packets from oWSN motes in the central area of the sensor field as well as oWSN motes on the outskirts of the sensor field.
- The packets were captured in such a manner that the fWSN did not interfere with the oWSN.

- The fWSN could be monitored on the fWSN management server in a real-time fashion and the logs were forensically saved in case they needed to be examined at a later stage.

It is concluded at this stage that the prototype successfully added a digital forensic readiness layer to an existing WSN, as the forensic data is available at any given moment should a forensic investigation need to be launched. It would only oblige the investigator to acquire the logs of the fWSN immediately upon the start of the investigation, and then to examine them.

Section 9.3 discusses demonstration II, which was used to show how flooding is dealt with by the prototype.

9.3 Demonstration II: Flooding Attacks on WSNs

This demonstration shows how the prototype fares in an environment where a flooding attack is launched against the oWSN. For this purpose, the environment is set up as can be seen in Figure 9.5.

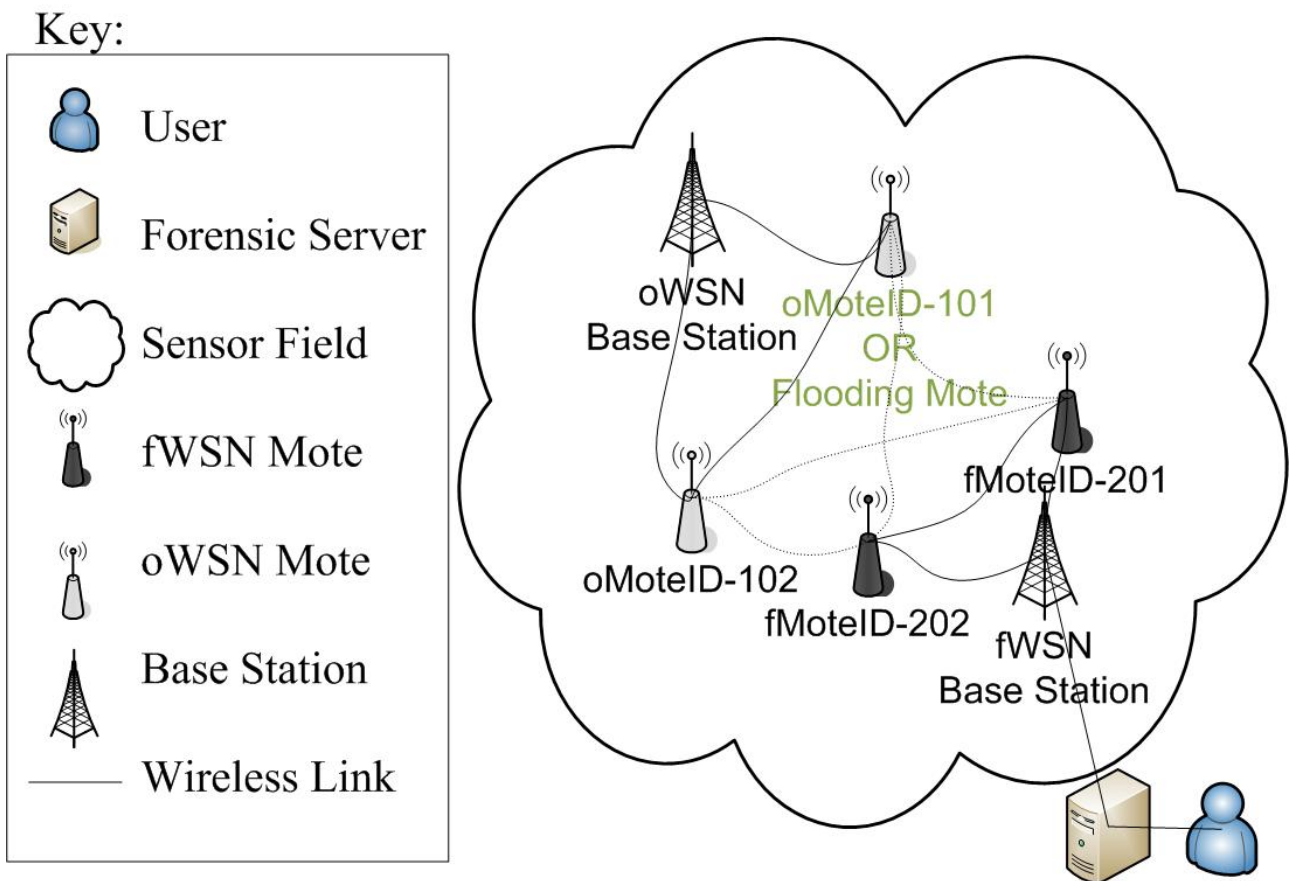
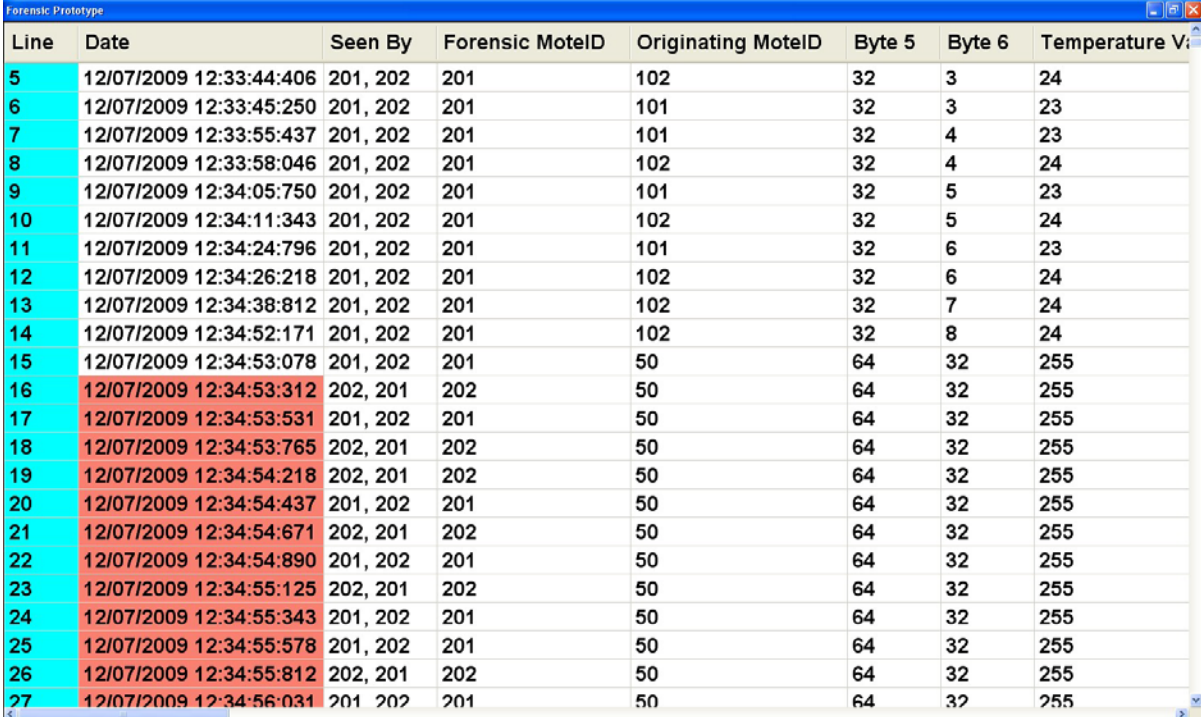


Figure 9.5. Demonstration II: Wireless Sensor Network Layout for Flooding Attack

The oWSN consists of one mote, oMoteID-102, and an oWSN base station. The fWSN consists of two fWSN motes – fMoteID-201 and fMoteID-202 – and an fWSN base station. The fWSN base station is directly connected to a management server. In Figure 9.5, oMoteID-101 is later replaced

with a mote designed to cause a flooding attack on a WSN. This flooding mote is in range of all three motes, namely oMoteID-101, fMoteID-201 and fMoteID-202.

Three screenshots from the packet analysis tool illustrate this demonstration. The first screen capture (Figure 9.6) illustrates the start of the flooding attack. The second screen capture (Figure 9.8) illustrates the effect the flooding has on the oWSN and fWSN. The third screen capture (Figure 9.10) illustrates what happens in the case that this was not a flooding attack and how the fWSN would have reacted.



Line	Date	Seen By	Forensic MoteID	Originating MoteID	Byte 5	Byte 6	Temperature V:
5	12/07/2009 12:33:44:406	201, 202	201	102	32	3	24
6	12/07/2009 12:33:45:250	201, 202	201	101	32	3	23
7	12/07/2009 12:33:55:437	201, 202	201	101	32	4	23
8	12/07/2009 12:33:58:046	201, 202	201	102	32	4	24
9	12/07/2009 12:34:05:750	201, 202	201	101	32	5	23
10	12/07/2009 12:34:11:343	201, 202	201	102	32	5	24
11	12/07/2009 12:34:24:796	201, 202	201	101	32	6	23
12	12/07/2009 12:34:26:218	201, 202	201	102	32	6	24
13	12/07/2009 12:34:38:812	201, 202	201	102	32	7	24
14	12/07/2009 12:34:52:171	201, 202	201	102	32	8	24
15	12/07/2009 12:34:53:078	201, 202	201	50	64	32	255
16	12/07/2009 12:34:53:312	202, 201	202	50	64	32	255
17	12/07/2009 12:34:53:531	201, 202	201	50	64	32	255
18	12/07/2009 12:34:53:765	202, 201	202	50	64	32	255
19	12/07/2009 12:34:54:218	202, 201	202	50	64	32	255
20	12/07/2009 12:34:54:437	201, 202	201	50	64	32	255
21	12/07/2009 12:34:54:671	202, 201	202	50	64	32	255
22	12/07/2009 12:34:54:890	201, 202	201	50	64	32	255
23	12/07/2009 12:34:55:125	202, 201	202	50	64	32	255
24	12/07/2009 12:34:55:343	201, 202	201	50	64	32	255
25	12/07/2009 12:34:55:578	201, 202	201	50	64	32	255
26	12/07/2009 12:34:55:812	202, 201	202	50	64	32	255
27	12/07/2009 12:34:56:031	201, 202	201	50	64	32	255

Figure 9.6. Demonstration II: The start of the flooding attack.

Figure 9.6 shows that the flooding attack started from line 15 onwards. The flooding attack was only detected from line 16, as one would never be able to determine from a single packet that it is a flooding attack. The “date” column is shaded in grey for those areas where the flooding attack has been detected.

In the demonstration, we have both oMoteID-102 and oMoteID-101 present in the oWSN for the first minute. After this minute, oMoteID-101 was removed from the oWSN and replaced with a flooding mote. This was done because the researcher was limited to having only 4 motes in total and thus had to replace oMoteID-101’s software with flooding software and redeploy it as a flooding mote. The flooding mote had been assigned a MoteID of 50 in order to improve the readability of the figures and to easily distinguish them from oMoteID-101. This flooding mote was set to transmit random data packets as fast as possible. The flooding mote was able to send about

six to seven data packets per second. This caused the sensor field to become saturated with data packets and could lead to the loss of data packets and rapid draining of mote batteries. In Figure 9.6 it can be seen that the packet analysis tool was able to detect almost instantly that there is flooding in the sensor field. Line 16 and onwards are shaded in grey to show that flooding was detected.

The packet analysis tool uses a simple algorithm to detect this flooding. Flooding is detected by means of the flooding coefficient that is defined and proposed by the researcher in the equation below. The flooding coefficient is calculated as shown in Equation 9.1.

Equation 9.1. Calculation of the flooding coefficient

$$f_k = \frac{f_{k-1} + y_k}{2}$$

where y_k is the amount of data packets received per time interval,

$$f_0 = 0 \text{ and } k \in \mathbb{R} \text{ (rational numbers) and } k \neq 0$$

This equation allows the prototype to detect if there is flooding in the network by checking for flooding at every time interval. The time interval can be defined in terms of seconds and merely determines how often the network is tested for flooding. For purposes of the prototype implementation, a time interval of ten seconds was used as this had been found to be an efficient interval without overloading the management server. It is important to note that the flooding coefficient is determined with the previous flooding coefficient being taken into account. This allows us to calculate on average how many more packets there are in the oWSN. After the flooding coefficient has been determined, the prototype determines how many more packets on average were received during this time interval. This number, defined by the researcher as T_k is the number of extra data packets received per time interval and is calculated using Equation 9.2.

Equation 9.2. Calculation of the number of extra data packets per time interval

$$T_k = f_k - f_{k-1} \text{ where } f_0 = 0 \text{ and } k \in \mathbb{R} \text{ and } k \neq 0$$

For the sake of this demonstration it was assumed that if T_k is larger than a set threshold that is slightly more than the number of data packets that could be expected if there are several retransmissions in the oWSN due to network errors in the oWSN, then the traffic will be flagged as flooding. The threshold used in this demonstration was ten packets per time interval, but it is important to note that this value should be determined by taking various factors of the oWSN into account. Ten packets were used in this research, because when one uses only two oWSN motes, it would be unlikely for them to have enough transmission errors to generate ten or more data packets per time interval.

The algorithm enables us to immediately detect flooding. Other advantages of the algorithm will be shown later in this demonstration. When flooding is detected by the fWSN management server, the following message is displayed.



Figure 9.7. Demonstration II: First flooding alert message

Figure 9.7 shows the alert message that appears on the management server when flooding is detected. This can be altered in many ways, for example one could let the management server send an e-mail to the WSN administrator. In this demonstration a time frame of one minute per flooding check was used. The message shows that in the previous time frame an average of 29 packets were received. In the current time frame 136 packets were received and thus flooding may have possibly occurred. This is flagged as flooding, because T_k would be equal to 107, which is more than our threshold of ten. The calculation of T_k is shown in Equation 9.3.

Equation 9.3. Calculation of the flooding in Figure 9.7

$$T_k = f_k - f_{k-1}$$

$$T_k = 136 - 29$$

$$T_k = 107$$

Flooding detection on its own has little to almost no use for the oWSN if the oWSN administrator does not act on the flooding warning. The oWSN administrator is required to physically investigate the flooding attack in the field. This task is made easier by the proposed prototype, because it can be used to make a rough estimate of where the flooding mote is located. Only fWSN motes that are within range of the flooding mote will be affected by the flooding mote and thus it is possible to determine the location of the flooding mote. Ideally the fWSN motes would be hand deployed and have their locations mapped. Having deployed the motes by hand allows one to determine the location of the flooding mote to within a few meters, depending on the radio strength of the fWSN motes. Otherwise one should be able to gather the position of the mote from the oWSN. This can be achieved if the oWSN was issued with a global positioning system (if suitable for the particular environment) in order to have the location of the motes mapped.

As far as the researcher is aware, there is currently no viable solution to detect flooding in WSNs. Hence it can be assumed that flooding detection in itself is already a big step forward in solving the flooding issue within WSNs. Figure 9.8 has been included to show the devastating effects that a flooding attack has on a WSN.

Line	Date	Seen By	Forensic MotelD	Originating MotelD	Byte 5	Byte 6	Temperature V
13	12/07/2009 12:34:38:812	201, 202	201	102	32	7	24
14	12/07/2009 12:34:52:171	201, 202	201	102	32	8	24
75	12/07/2009 12:35:08:531	201, 202	201	102	32	9	24
188	12/07/2009 12:35:36:234	201, 202	201	102	32	11	24
244	12/07/2009 12:35:49:859	201	201	102	32	12	24
295	12/07/2009 12:36:03:031	201, 202	201	102	32	13	24
408	12/07/2009 12:36:30:718	202	202	102	32	15	24
517	12/07/2009 12:36:57:296	202	202	102	32	17	24
571	12/07/2009 12:37:10:921	201, 202	201	102	32	18	24
683	12/07/2009 12:37:38:171	201	201	102	32	20	24
734	12/07/2009 12:37:51:109	202	202	102	32	21	24
790	12/07/2009 12:38:05:187	201, 202	201	102	32	22	24
1162	12/07/2009 12:39:39:421	201	201	102	32	29	24
1219	12/07/2009 12:39:53:953	202	202	102	32	30	24
1370	12/07/2009 12:40:46:328	201	201	102	32	34	24
1426	12/07/2009 12:41:00:406	201	201	102	32	35	24
1481	12/07/2009 12:41:13:968	201	201	102	32	36	24
1595	12/07/2009 12:41:41:453	202	202	102	32	38	24
15	12/07/2009 12:34:53:078	201, 202	201	50	64	32	255
16	12/07/2009 12:34:53:312	202, 201	202	50	64	32	255
17	12/07/2009 12:34:53:531	201, 202	201	50	64	32	255
18	12/07/2009 12:34:53:765	202, 201	202	50	64	32	255
19	12/07/2009 12:34:54:218	202, 201	202	50	64	32	255

Figure 9.8. Demonstration II: Impact of flooding on WSNs

In Figure 9.8 we can see that several data packets were lost during the flooding attack. This occurs when the flooding mote and any other mote attempt to transmit a data packet at exactly the same time and these motes collide. Line 244, line 408, line 517, line 683 and line 734 are just some examples of where data packets were lost due to the flooding attack, as the packets were captured by only one of the fWSN motes. This shows that the flooding attack caused a DoS attack that affected both the oWSN and the fWSN, but due to the nature of the fWSN (which has built-in redundancy) the data packets were still captured. The built-in redundancy is due to the fact that the fWSN has several motes within range of the flooding attack; thus, if one of the fWSN motes misses a flooding data packet, the other motes will still detect the flooding attack. In the “Byte 6” column (the sequence number field for the oWSN) we can see that the sequence numbers do not necessarily follow chronologically. This is also due to the flooding attack.

The prototype is in itself not resilient against the flooding attack, as it also suffers some noticeable packet loss. The researcher does not foresee this to be an issue, as the flooding mote should be eliminated as quickly as possible by the oWSN administrator to ensure that very little

harm is caused by the flooding attack. This would also cause the flooding attack to be eliminated from the fWSN, once it has been eliminated from the oWSN. As far as the researcher is aware, there is also currently no communication protocol for WSNs that are resilient against flooding attacks. In the case where a resilient communication protocol becomes available, it can easily be added to the prototype implementation as we have used the most basic communication protocol for the prototype currently available.

There is, however, the instance where the flooding coefficient can cause the threshold to be exceeded under normal operation of the oWSN. One such possible scenario would be if additional oWSN motes were added while the fWSN motes were in operation. In order to simulate this scenario, the flooding mote was left in the oWSN. In Equation 9.1 the previous flooding coefficient is used together with the new y_k value that is determined after each elapsed time frame. This would cause the T_k value to gradually decrease, as the difference between the number of data packets received in each time frame would gradually decrease. This phenomenon can be noted by examining Figure 9.9.

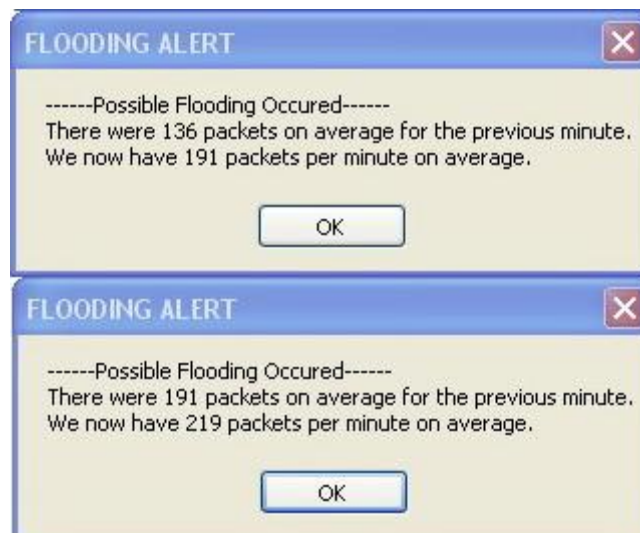


Figure 9.9. Demonstration II: Continuous flooding alerts

In Figure 9.9 one can see that the value of T_k , which is the difference between the number of data packets received in the current time frame and the number of data packets received in the previous time frame, will continuously decrease until it drops below the predefined threshold value. For the first box in Figure 9.9 one can determine (by using Equation 9.2) that the value of T_k would be 55 while in the second box the value of T_k would be 28. This causes the fWSN management server to go into a state where the T_k value will be below the threshold, as it will have learnt over time that it is supposed to see more packets per time frame. This event is illustrated in Figure 9.10.

Line	Date	Seen By	Forensic MotelD	Originating MotelD	Byte 5	Byte 6	Temperature V:
1235	12/07/2009 12:39:57:828	201, 202	201	50	64	32	255
1236	12/07/2009 12:39:58:046	202, 201	202	50	64	32	255
1237	12/07/2009 12:39:58:281	201, 202	201	50	64	32	255
1238	12/07/2009 12:39:58:500	201, 202	201	50	64	32	255
1239	12/07/2009 12:39:58:734	202, 201	202	50	64	32	255
1240	12/07/2009 12:39:59:187	201, 202	201	50	64	32	255
1241	12/07/2009 12:39:59:406	202, 201	202	50	64	32	255
1242	12/07/2009 12:39:59:640	201, 202	201	50	64	32	255
1243	12/07/2009 12:39:59:859	202, 201	202	50	64	32	255
1244	12/07/2009 12:40:00:312	201, 202	201	50	64	32	255
1245	12/07/2009 12:40:00:781	201, 202	201	50	64	32	255
1246	12/07/2009 12:40:01:000	202, 201	202	50	64	32	255
1247	12/07/2009 12:40:01:031	201, 202	201	50	64	32	255
1248	12/07/2009 12:40:01:234	201, 202	201	50	64	32	255
1249	12/07/2009 12:40:01:687	202, 201	202	50	64	32	255
1250	12/07/2009 12:40:02:140	202, 201	202	50	64	32	255
1251	12/07/2009 12:40:02:359	201, 202	201	50	64	32	255
1252	12/07/2009 12:40:02:500	201, 202	201	50	64	32	255
1253	12/07/2009 12:40:02:828	202, 201	202	50	64	32	255
1254	12/07/2009 12:40:03:046	201, 202	201	50	64	32	255
1255	12/07/2009 12:40:03:281	202, 201	202	50	64	32	255
1256	12/07/2009 12:40:03:500	201, 202	201	50	64	32	255
1257	12/07/2009 12:40:03:734	202, 201	202	50	64	32	255

Figure 9.10. Demonstration II: End of the flooding due to T_k being below the threshold

Figure 9.10 shows that all traffic after the 12:40 minute mark was not flagged as flooding anymore. Hence the fWSN management server was able to learn over time that the fWSN administrator deemed the extra traffic as normal and would not flag it as flooding anymore. It should not be argued that this should still be flagged as flooding, as one would have had ample prior warning of the flooding. However, if one chooses to ignore the flooding warnings until this event occurs, it will be the fault of the oWSN administrator and not a fault of the software.

In this section the demonstration has shown that the proposed prototype was successfully able to detect flooding in a WSN environment. The prototype provided the oWSN administrator with information on when the flooding occurred and roughly in which area of the oWSN sensor field the flooding occurred, by showing which fWSN motes were able to detect the flooding mote. It was then up to the oWSN administrator to physically go and eliminate the flooding mote.

The prototype also proved to be resilient against detecting false cases of flooding. The prototype can learn over time that an increase in the number of data packets captured is normal in the case that there is not really a flooding attack but simply an influx of data packets, which could be due to more oWSN nodes introduced in the field. The prototype will therefore not cause false-positives after it has learnt such information. If the traffic returns to normal again after a flooding mote was eliminated, the number of data packets per time frame would decrease accordingly and the prototype would again adjust the flooding detection accordingly.

The following section is devoted to showing how the prototype can be used to verify sensory data received by the oWSN. The demonstration also shows how the integrity and authenticity of the data packets captured can be determined by matching the sensory data with the sensory data from the oWSN.

9.4 Demonstration III: Sensory Data Verification

This demonstration was done to show that the data packets captured by the prototype retain their authenticity and integrity. The environment was set up as shown in Figure 9.11.

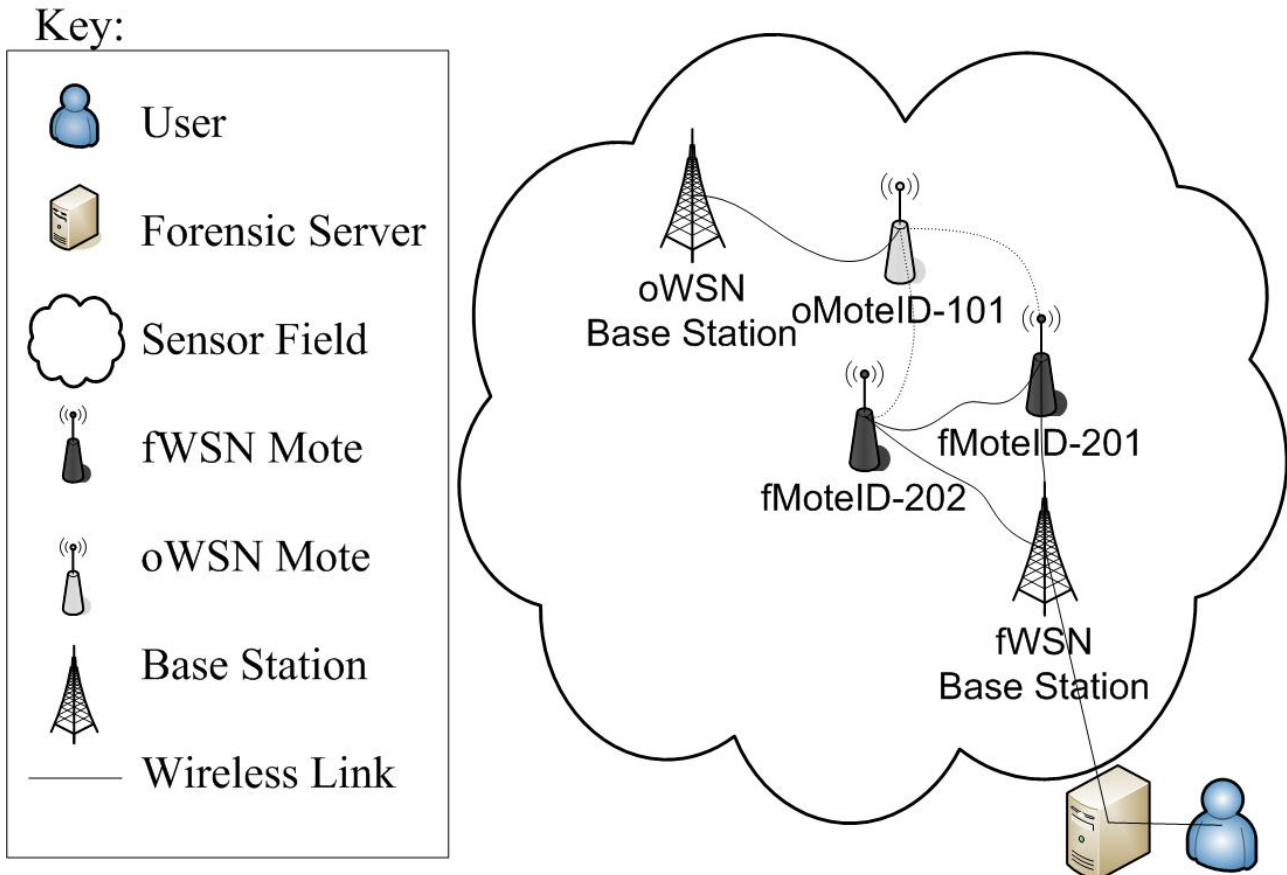


Figure 9.11. Demonstration III: Wireless Sensor Network Layout for Sensory Data Verification

The oWSN consisted of a single mote, namely oMoteID-101, and an oWSN base station. The fWSN consisted of two motes, fMoteID-201 and fMoteID-202, as well as an fWSN base station and an fWSN management server that was connected directly to the fWSN base station. At all times throughout the demonstration oMoteID-101 was within range of both fMoteID-201 and fMoteID-202. At specific intervals throughout the demonstration the physical location of oMoteID-101 was altered to achieve specific light and temperature conditions. The location of oMoteID-101 was altered to show that the fWSN revealed the difference in sensory data that can be seen from the data packets having originated from the oWSN.

Figure 9.12 provides a screen capture of the packet analysis tool. The screen capture shows a five-minute time frame of all the data packets in the oWSN.

Line	Date	Seen By	Forensic MotelID	Byte 1	Byte 2	Originating MotelID	Byte 4	Byte 5	Byte 6	Byte 7	Light Value	Tempe Value
4	12/07/2009 12:52:08:734	202, 201	202	126	0	101	255	32	4	0	24	23
5	12/07/2009 12:52:18:812	202, 201	202	126	0	101	255	32	5	0	24	24
6	12/07/2009 12:52:29:234	202, 201	202	126	0	101	255	32	6	0	24	25
7	12/07/2009 12:52:39:625	202, 201	202	126	0	101	255	32	7	0	24	26
8	12/07/2009 12:52:49:812	202, 201	202	126	0	101	255	32	8	0	24	27
9	12/07/2009 12:53:00:062	202, 201	202	126	0	101	255	32	9	0	24	29
10	12/07/2009 12:53:10:515	202, 201	202	126	0	101	255	32	10	0	24	30
11	12/07/2009 12:53:20:828	202, 201	202	126	0	101	255	32	11	0	24	31
12	12/07/2009 12:53:30:890	202, 201	202	126	0	101	255	32	12	0	24	31
13	12/07/2009 12:53:41:328	202, 201	202	126	0	101	255	32	13	0	24	32
14	12/07/2009 12:53:51:703	202, 201	202	126	0	101	255	32	14	0	24	32
15	12/07/2009 12:54:01:921	202, 201	202	126	0	101	255	32	15	0	24	33
16	12/07/2009 12:54:12:187	202, 201	202	126	0	101	255	32	16	0	24	33
17	12/07/2009 12:54:22:406	202, 201	202	126	0	101	255	32	17	0	24	34
18	12/07/2009 12:54:32:859	202, 201	202	126	0	101	255	32	18	0	24	35
19	12/07/2009 12:54:43:000	202, 201	202	126	0	101	255	32	19	0	24	35
20	12/07/2009 12:54:53:328	202, 201	202	126	0	101	255	32	20	0	24	36
21	12/07/2009 12:55:03:687	202, 201	202	126	0	101	255	32	21	0	24	36
22	12/07/2009 12:55:14:156	202, 201	202	126	0	101	255	32	22	0	0	33
23	12/07/2009 12:55:24:265	202, 201	202	126	0	101	255	32	23	0	0	32
24	12/07/2009 12:55:34:546	202, 201	202	126	0	101	255	32	24	0	1	30
25	12/07/2009 12:55:44:921	202, 201	202	126	0	101	255	32	25	0	0	29
26	12/07/2009 12:55:55:328	202, 201	202	126	0	101	255	32	26	0	0	28
27	12/07/2009 12:56:05:453	202, 201	202	126	0	101	255	32	27	0	0	27
28	12/07/2009 12:56:15:796	202, 201	202	126	0	101	255	32	28	0	1	26
29	12/07/2009 12:56:26:140	202, 201	202	126	0	101	255	32	29	0	0	25
30	12/07/2009 12:56:36:484	202, 201	202	126	0	101	255	32	30	0	0	24
31	12/07/2009 12:56:46:656	202, 201	202	126	0	101	255	32	31	0	0	24
32	12/07/2009 12:56:57:140	202, 201	202	126	0	101	255	32	32	0	1	23
33	12/07/2009 12:57:07:265	202, 201	202	126	0	101	255	32	33	0	0	22
34	12/07/2009 12:57:17:703	202, 201	202	126	0	101	255	32	34	0	0	22

Figure 9.12. Demonstration III: Packet analysis tool screen capture to show the change in light and temperature throughout the oWSN

The screen capture in Figure 9.12 shows a five-minute time frame of the demonstration. At 12:51 when the oWSN was switched on, oMoteID-101 was positioned in an environment with average lighting and at room temperature (about 23°C). At this time the fWSN was also switched on and started capturing data packets from oMoteID-101. At time stamp 12:52, the oMoteID-101 was moved into an environment that had direct sunlight. This environment can be seen in Figure 9.13.



Figure 9.13. Demonstration III: oMoteID-101 in a direct sunlight environment

oMoteID-101 was left in direct sunlight for two minutes, after which it was placed into a refrigerator shortly after 12:55, where there was less light and lower temperatures occurred. This environment can be seen in Figure 9.14.

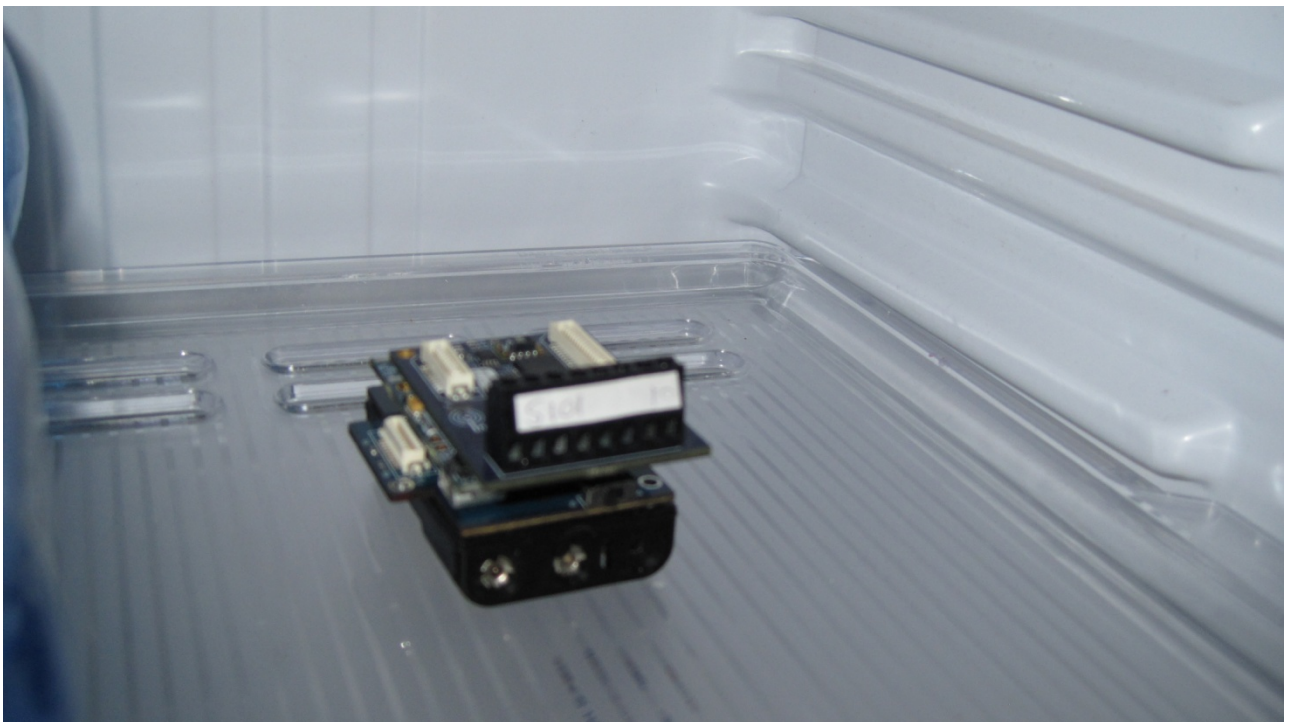


Figure 9.14. Demonstration III: oMoteID-101 placed in a refrigerator

oMoteID-101 was left in the refrigerated environment until 12:58, at which time the demonstration was terminated. The screen capture in Figure 9.12 only depicts the time frame 12:52 until 12:58 and thus the further discussion focuses only on this time frame.

In Figure 9.12 the “Light Value” column depicts the lighting conditions that the sensor measured. These values range from 0 up to 24. A lower value depicts little to no direct light, whereas a higher value depicts a high concentration of direct light. The “Temperature Value” column depicts the temperature reading of the temperature sensor that is present on the iMote2 sensor board. Throughout Figure 9.12 one can see that the temperature and light values in the log show different trends during different time frames. These different trends will be discussed next, while keeping in mind the timeline when the location of oMoteID-101 was altered.

At 11:51 oMoteID-101 and the fWSN were switched on. The readings for these times are not visible in Figure 9.12 and thus the discussion will proceed from time stamp 11:52. At 11:52 the oMoteID-101 was moved into direct sunlight, which caused the temperature reading to increase as the oMoteID-101 was heating up. As can be seen in the screen capture in Figure 9.12, the temperature gradually increased on every data packet transmission from oMoteID-101. The light value stayed at a constant 24 as the mote was exposed to direct sunlight throughout the entire time. At 12:55:05 oMoteID-101 was moved into a refrigerated environment and the refrigerator door was closed. The ambient temperature of the refrigerator was about 14°C and while the refrigerator door was closed, little or no light could enter the refrigerator. This caused the reading on the sensors on oMoteID-101 to decrease. The light sensor measured an immediate change in lighting conditions and altered to either a 0 or 1 from line 22 onwards. The temperature sensor started to gradually cool down while oMoteID-101 was kept in the refrigerator. This caused the temperature reading to gradually decrease. As previously mentioned, the demonstration was terminated at 12:58, as it merely intended to show that the fWSN was able to record data packets from the oWSN while retaining integrity and authenticity (and not to demonstrate the ability of WSNs to measure light and temperature).

Throughout this demonstration one could see that the data packets captured by the fWSN reflected data that was true to the environment in which the oWSN resided. A further validation of the data packets can be done by comparing the data logged by the fWSN to the data logged on the oWSN base station. This is not illustrated in this dissertation as it would be a simple one-to-one mapping of the data. A decision was made to rather illustrate this in a more practical way. The researcher decided to make use of the temperature sensor and show that when the oWSN mote was put in a place where the temperature would be altered, then this change would be evident from the

fWSN data. The usage of temperature sensor data, as shown in Figure 9.12, allows one to verify the fWSN data by examining the fact that the fWSN also shows the exact same fluctuation in temperature as was seen in the oWSN.

The notion that the fWSN is able to capture all of the data packets travelling through the oWSN has many advantages. Since it has now been proved by means of demonstration that data packets arriving at the fWSN management server reflect the true data packets that were transmitted in the oWSN, we can use this information to verify data received by the oWSN base station. Also, if any harm or data loss occurs at the oWSN base station, the fWSN log can be used to fill in the missing data of the oWSN. Again, this is not invasive of the oWSN and simply offers a feature that could be utilised by the oWSN administrator if data has gone missing on the oWSN management server itself. The fWSN data log can also be used to perform a simulation of the events that occurred in the oWSN by attempting to replay the data packets through a simulated WSN to determine the network topology or network events that occurred to cause a certain incident. It should however be noted that this might be a very expensive exercise. The notion of reusing the fWSN data log is discussed in more detail later in this dissertation.

A discussion of the fourth and final demonstration follows next. It shows how the prototype is able to detect any form of communication that occurs within its frequency range.

9.5 Demonstration IV: Capturing of TelosB data packets

Demonstration IV shows how the prototype is able to listen for data packets in a promiscuous mode. For this demonstration, the environment is set up as illustrated in Figure 9.15.

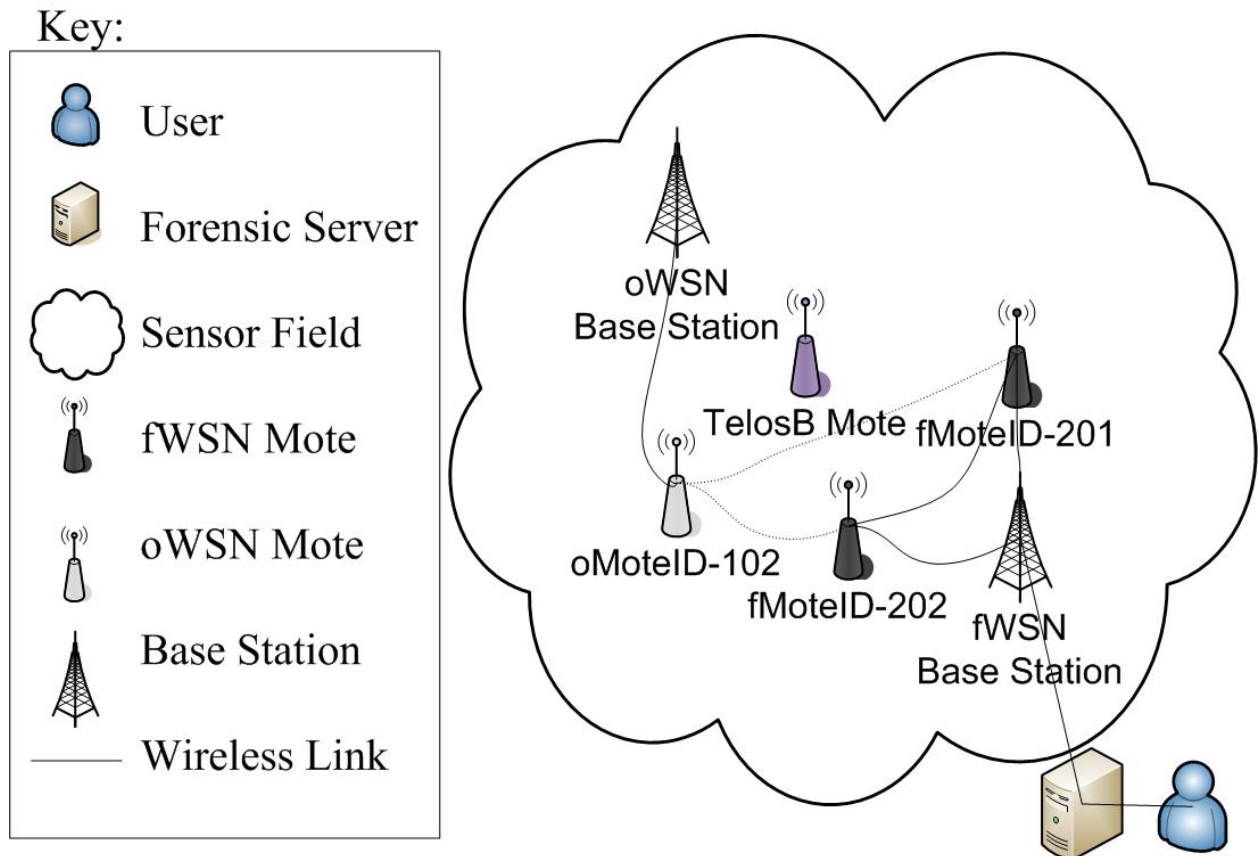


Figure 9.15. Demonstration IV: Capturing TelosB Data Packets

In the demonstration the oWSN consists of a single mote, namely oMoteID-102, and an oWSN base station. Yet again the demonstration will be using the full fWSN consisting of two motes, an fWSN base station and the fWSN management server. The TelosB mote, which was introduced in Section 7.3, will also be used as part of this demonstration. As mentioned earlier, the TelosB mote (see Figure 9.16) had predefined software deployed onto it and the researcher had no knowledge as to what the use of this software was.



Figure 9.16. Demonstration IV: Crossbow TelosB mote with predefined software

After switching on the TelosB mote, it was determined that if you press the user button on the mote it will transmit a data packet. This data packet was transmitted on an unknown frequency. All that the researcher knew about the TelosB mote was that it is only capable of communicating within the capable frequency ranges that the iMote2s can handle (i.e. 2.4GHz to 2.4835GHz) (Crossbow Technology Inc, 2007; Crossbow Technology Inc, 2005). It was for this very reason that the researcher decided to execute this demonstration with a TelosB mote of which he had no previous knowledge in terms of the software deployed on it and what the purpose of the mote was. He knew that there was some kind of operational software on the mote, but was also unaware of what the data transmitted by the TelosB mote resembles. For this reason, the columns in Figure 9.17 were kept with their original labels, namely from “Byte 1” to “Byte 64”.

Line	Date	Seen By	Forensic MoteID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
3	12/07/2009 13:02:22:484	201, 202	201	126	0	47	125	31	18	0	193	27	158
5	12/07/2009 13:02:42:328	201, 202	201	126	0	47	125	31	18	0	192	75	10
7	12/07/2009 13:03:00:500	201, 202	201	126	0	47	125	31	18	0	192	22	241
11	12/07/2009 13:03:29:046	201, 202	201	126	0	47	125	31	18	0	192	11	107
13	12/07/2009 13:03:47:046	201, 202	201	126	0	47	125	31	18	0	192	86	21
16	12/07/2009 13:04:11:453	201, 202	201	126	0	47	125	31	18	0	192	65	42
18	12/07/2009 13:04:29:984	201, 202	201	126	0	47	125	31	18	0	192	207	139
22	12/07/2009 13:06:09:437	201, 202	201	126	0	47	125	31	18	0	192	50	156
23	12/07/2009 13:06:37:000	201, 202	201	126	0	47	125	31	18	0	193	102	172
1	12/07/2009 13:02:07:718	201, 202	201	126	0	102	255	32	1	0	54	32	22
2	12/07/2009 13:02:21:390	201, 202	201	126	0	102	255	32	2	0	59	28	22
4	12/07/2009 13:02:34:843	201, 202	201	126	0	102	255	32	3	0	20	27	22
6	12/07/2009 13:02:48:453	201, 202	201	126	0	102	255	32	4	0	243	27	22
8	12/07/2009 13:03:02:000	201, 202	201	126	0	102	255	32	5	0	255	27	22
9	12/07/2009 13:03:15:359	201, 202	201	126	0	102	255	32	6	0	237	27	22
10	12/07/2009 13:03:28:796	201, 202	201	126	0	102	255	32	7	0	56	27	22
12	12/07/2009 13:03:42:406	201, 202	201	126	0	102	255	32	8	0	50	27	22
14	12/07/2009 13:03:55:890	201, 202	201	126	0	102	255	32	9	0	54	26	22
15	12/07/2009 13:04:09:312	201, 202	201	126	0	102	255	32	10	0	54	26	22
17	12/07/2009 13:04:22:890	201, 202	201	126	0	102	255	32	11	0	49	26	22
19	12/07/2009 13:04:36:437	201, 202	201	126	0	102	255	32	12	0	52	26	22
20	12/07/2009 13:04:49:859	201, 202	201	126	0	102	255	32	13	0	51	26	22
21	12/07/2009 13:05:03:328	201, 202	201	126	0	102	255	32	14	0	49	26	23

Figure 9.17. Demonstration IV: TelosB communication captured by the fWSN

In Figure 9.17 it is important to note that the bottom half of the figure (from where the “Byte 3” column is equal to 102) represents data from oMoteID-102. This is because the oMoteID-102 was also present in the field during experimentation with the TelosB mote. The “Byte 3” column may thus be assumed to represent the “Originating MoteID” in Figure 9.17.

The top half of the figure shows the data captured from the TelosB mote. The following lines represent the data packets from the TelosB mote: Line 3, 5, 7, 11, 13, 16, 18, 22 and 23. The fWSN was able to forensically capture data packets from the TelosB mote without having any knowledge of the communication protocol used by the TelosB mote. The fWSN simply relied on the notion that the TelosB can only communicate in the same frequency range on which the fWSN is able to communicate. This proves that the proposed prototype is able to capture data packets from various types of WSN equipment, as long as the frequency range of these devices is similar to the WSN equipment on which the prototype is implemented.

This demonstration also indicates that different types of WSN equipment can be used in the oWSN and the prototype will still be able to forensically capture data packets from the oWSN. This confirms that if the prototype is deployed on WSN equipment already existing in your network, that the prototype will be able to function in your implementation of the network.

It is not possible for the researcher to fully explain the data packets captured from the TelosB mote. This is because he had no previous knowledge of the use of the TelosB mote. In order to fully understand the data captured, one would have to know the function of the TelosB mote. This will be the case in any environment where the fWSN is deployed. It can, however, be safely assumed that if

one deploys the fWSN into an oWSN you would already be familiar with the basic function of the oWSN. The data packets captured on the fWSN can also be replayed through a simulated WSN in order to simulate what the use of the oWSN was.

The following section concludes this chapter and provides a brief overview of the prototype.

9.6 Conclusion

This chapter fully demonstrated the prototype and serves as proof that the prototype is viable in various real-life WSN implementations. This is clear from examining all the various demonstrated examples. The first demonstration showed evidence of how the prototype was able to function in most real-life WSN implementations. The second demonstration provides evidence that the prototype was able to successfully detect flooding attacks in a WSN environment. The third demonstration showed how the data packets captured from the fWSN retained their authenticity and integrity and how these data packets can be used for further analysis of the oWSN. The final demonstration showed that the fWSN was able to forensically capture data packets from various types of WSN devices as long as these devices communicated within the same frequency range as the fWSN.

Part V concludes the dissertation by examining how this prototype can be used to improve current WSN implementations.

Part V: Conclusions

Chapter 10 **Discussions on Digital Forensic Readiness and Flooding Detection**

10.1 Introduction

This chapter highlights the three main contributions that this study has made. It also discusses why these items should be considered contributions to the field of WSNs. For this reason, this chapter is divided into three components that focus on the three contributions, namely the WSN digital forensic readiness requirements, the WSN digital forensic readiness model and the WSN digital forensic readiness prototype.

10.2 WSN digital forensic readiness requirements

This dissertation provided a list of WSN digital forensic readiness requirements in Chapter 5. It was important to start off by proposing this list, as the only publication to define a list of requirements in the literature on WSNs is the one by Mouton and Venter (2011). Their publication contains the exact same list as is proposed in this dissertation.

Digital forensic readiness in itself is still a relatively new field and has only been around since the start of the 21st century. It is most likely because of this and the fact that WSNs is also a fairly new concept that these two fields have yet to be linked together. The researcher consequently decided that it would be appropriate to define a list of WSN digital forensic readiness requirements.

The list was created by taking into account various digital forensic readiness requirements from other domains. This dissertation simply made these digital forensic readiness requirements more specific to WSNs. It has shown WSNs to be a fairly unique field that has some very specific requirements, which is not the case with ordinary IEEE 802.11 wireless networks.

Having now formally proposed this list in Chapter 5, it will be easier for other researchers to attempt to implement digital forensic readiness on WSN devices. All of the currently known special requirements that should be taken into account while working with WSN devices have also been included in this list. This list can therefore be used as a checklist of requirements to adhere to when attempting to propose any specific digital forensic readiness model into a WSN domain.

By providing a WSN digital forensic readiness requirements list, this study has undoubtedly made an important contribution to the WSN domain. The requirements list was also put to the test as the model proposed in this dissertation was based on it.

The following section examines the WSN digital forensic readiness model proposed in this dissertation.

10.3 WSN Digital forensic readiness model

The WSN digital forensic readiness model that was proposed in Chapter 6 adheres to all the WSN digital forensic readiness requirements proposed in Chapter 5.

The model was designed in such a way that it can be implemented on an existing WSN without any modification to this WSN. This decision was made so that it would be easier for people to adopt this model as it would have no negative influence on their existing WSN. People often tend to prefer an “out-of-the-box” solution to something they would have to go and additionally implement on the equipment along with their other software.

This study has shown that the WSN digital forensic readiness requirements can be used as a way to define a WSN digital forensic readiness model. After the model was defined, we used the WSN digital forensic readiness requirements in order to evaluate the WSN digital forensic readiness model. By evaluating the WSN digital forensic readiness model in this way, the researcher showed that the WSN digital forensic readiness requirements were of great use in the design of a WSN digital forensic readiness model. It also provided an easy way to determine if it was at all viable to implement the WSN digital forensic readiness model in a WSN domain.

The model was also from the start intended to be both a digital forensic readiness solution and a means of detecting flooding in a WSN. These notions are discussed in the following section which deals with the WSN digital forensic readiness prototype.

10.4 WSN digital forensic readiness prototype

The WSN digital forensic readiness prototype, which was implemented on the iMote2 devices according to the specifications set out by the WSN digital forensic readiness model, had two main benefits to WSN devices. These two benefits are the ability to provide digital forensic readiness to WSN devices and the ability to detect flooding in a WSN. These two benefits are discussed separately in the two subsections below.

10.4.1 Digital forensic readiness

Providing digital forensic readiness to WSN devices was the main goal of this study, and this dissertation claims that the prototype was successfully able to do this. Throughout the demonstrations several factors were considered to test the feasibility of the prototype in real-life WSN applications.

The first demonstration focused on testing the prototype in an environment that is most likely to be found in an existing WSN environment. The prototype considered what would happen if there

were oWSN motes in the centre and also on the outskirts of the sensor field. The demonstration showed that the authenticity and integrity of the data packets were retained throughout the entire capturing of data packets by the fWSN. The fWSN could also be deployed after the oWSN had been switched on. This was demonstrated by only switching the fWSN on after the oWSN had been in operation for a while. The oWSN was furthermore tested to ensure that the fWSN did not cause any extra network overhead to the oWSN. This was accomplished by allowing the fWSN to communicate only on frequencies that were not used by the oWSN. Throughout this demonstration the prototype was able to successfully and forensically capture data packets from the oWSN. In order to prove to the reader that the data being captured was indeed forensically sound, the third demonstration was conducted. (The second demonstration will only be discussed in the next section, as it deals with flooding specifically and the entire section 10.4.2 is dedicated to flooding.)

The third demonstration involved a graphic illustration of where the oWSN mote was moved to a location where it received direct sunlight and then to a location where it received little or no direct light. This caused the oWSN mote to have different sensor readings, depending on its specific location. The demonstration showed that the fWSN was successfully able to capture the data packets reflecting these changes. This study also claims that the data packets which we now know were captured in a forensically sound fashion can be used to simulate the oWSN in a later investigation. The only other feature of the prototype in terms of digital forensic readiness was its ability to capture data packets from other WSN devices. This was demonstrated next.

The fourth demonstration introduced a TelosB mote into the sensor field and showed that the fWSN was able to capture data packets from the TelosB while also capturing data packets from the oWSN. This finding confirms the researcher's claim that any communication on the same frequency range as that on which the fWSN devices can communicate, will be forensically captured by the fWSN.

After considering all these demonstrations and the attributes of the fWSN illustrated, it is asserted that the prototype successfully illustrated the feasibility of the WSN forensic readiness model. In turn, it also illustrated the feasibility of the WSN forensic readiness requirements as set out in the dissertation.

The next section focuses on the other aspect of the prototype, namely its requirement to do flooding detection in a WSN domain.

10.4.2 Flooding detection

Flooding remains one of the important issues to consider within WSNs. At the time of writing this dissertation, no solution has yet been found to eliminate flooding attacks in a WSN domain. Since flooding attacks constitute such an important challenge, this study also focused on how the prototype can be used to detect flooding in the WSN domain. It is important to remember that the main goal of the prototype is to add a digital forensic readiness layer to an existing WSN and not at all to solve the flooding issue in WSNs. The latter was an opportune spin-off from the main goal of the study and the researcher considered it important enough to be included in the findings.

The second demonstration in Section 9.3 was aimed at illustrating how flooding attacks can be detected in WSNs. A flooding mote was introduced into the existing sensor field where the oWSN and fWSN were already residing. The fWSN management server was successfully able to detect the flooding attack. This server detected the flooding attack based on the value of the flooding coefficient proposed earlier. In our demonstration, the flooding attack was detected within the first minute that it occurred. It was nonetheless still up to the oWSN administrator to react on the flooding attack.

The other side of the case was also considered, namely that the increased number of data packets in the sensor field was a normal phenomenon. If the increased number of data packets was legitimate, the fWSN management server would still flag it as flooding for the first few timeframes. However, once a few of these timeframes have elapsed, the fWSN management server would have learnt that the increased number of data packets in the WSN is considered normal by the oWSN administrator and thus it will stop flagging the data packets as flooding packets.

In a case where flooding is incorrectly classified, this does not hamper the digital forensic readiness viability of the prototype, as only a message is provided to the oWSN administrator. The oWSN then has the choice to react to it or to ignore it. Regardless of the choice by the oWSN administrator, the fWSN will continue to capture data packets from both the flooding mote and the oWSN. This allows the functionality of flooding detection to be a feature of the prototype and not something the prototype specifically relies on. As a result, the flooding detection feature could simply be removed if it was not required in the WSN domain.

Section 10.5 concludes this chapter by showing how the three contributions are helpful to the WSN domain.

10.5 Conclusion

This chapter was devoted to discussing the three contributions made in this research. The contributions built on and thus reinforced one another, as each of them showed that they could be viable contributions to the WSN domain.

The WSN digital forensic readiness requirements were required to verify the WSN digital forensic readiness model. These requirements were also created because, at the time of writing this dissertation, no list of WSN digital forensic readiness requirements existed yet. A WSN digital forensic readiness model was developed to show that these WSN digital forensic readiness requirements were helpful.

The WSN digital forensic readiness model took all the WSN digital forensic readiness requirements into account. After the model had been proposed, it was yet again measured against the WSN digital forensic readiness requirements. This was done in order to show both the viability of the model and the requirements that were proposed.

In an attempt to fully illustrate the viability of the model, it was developed into a WSN digital forensic readiness prototype. The prototype was put up in four demonstration environments to show its viability in different environments. In each demonstration the prototype indeed proved to be successful in capturing the oWSN data packets in a forensically sound manner.

In the final chapter all three of these contributions are combined and their influence on each other is discussed.

Chapter 11 Conclusion

This study focused on how to add a digital forensic readiness layer to an existing WSN environment. In order to accomplish this, several factors had to be taken into account.

Since no set of WSN digital forensic readiness requirements had previously been defined in WSN literature, the researcher started to define a list of relevant requirements. The requirements were formulated by examining digital forensic readiness requirements from other domains and combining them with WSN-specific issues. The WSN-specific issues included items that were found in WSN devices only. These issues mostly concerned the way WSN devices communicate with each other, and the power and processing constraints of WSN device.

Once the WSN digital forensic readiness requirements had been identified, the dissertation proposed a WSN digital forensic readiness model. This model was implemented on a separate WSN that would afterwards be deployed into a pre-existing WSN. This feature in itself provided many beneficial attributes of the model. The model was subsequently measured against the WSN digital forensic readiness requirements in order to confirm its feasibility. The model adhered to all the WSN digital forensic readiness requirements and was subsequently developed into a WSN digital forensic readiness prototype.

All of the research above was conducted to answer the main research question: *Is it possible to implement a digital forensic readiness system for an existing wireless sensor network without relying on the existing wireless sensor network or any type of specific configuration that the existing wireless sensor has?* This question was answered by providing a prototype that was able to provide digital forensic readiness to an existing WSN. An unexpected spinoff of the research was the fact that the prototype also proved to have the ability to perform flooding detection – this was later found to be a very challenging part of WSNs, even though this had not actually been mentioned in the original problem statement.

Thus, the value of the prototype was twofold – it could provide a digital forensic readiness layer over an existing WSN and perform flooding detection in an existing WSN. The goal of digital forensic readiness was merely to facilitate the digital forensic investigation process and it did not form part of the actual investigation process. Firstly, the viability of a digital forensic readiness solution was tested. It was found that the prototype was able to capture data packets from an oWSN in a forensically sound manner without compromising the authenticity and integrity of the data packets. The data packets were then all stored on the fWSN management server and could be used afterwards to facilitate the digital forensic investigation process. The data captured could even be

used in a simulation environment to illustrate how the oWSN operated. Secondly, the feasibility of flooding detection by the prototype was tested. This was done by deploying a flooding mote into the sensor field while the fWSN was in operation. The fWSN management server was at all times successfully able to classify the attacks as flooding attacks. Nevertheless, the prototype still relied on the oWSN administrator to eliminate the flooding. Because the prototype was able to provide the oWSN administrator with the general location of the flooding, it was expected to be not too difficult to physically eliminate the flooding mote.

Having found that the prototype provides viable results in terms of adding a digital forensic readiness layer to an existing WSN, we could draw a number of conclusions:

- The WSN digital forensic readiness requirements should be taken into account when attempting to examine digital forensic readiness in a WSN environment. This is because both the model and prototype were designed by taking into account the WSN digital forensic readiness requirements.
- The proposed WSN digital forensic readiness model is a viable WSN digital forensic readiness model, because it adheres to the WSN digital forensic requirements. The feasibility of the model was shown by its successful conversion into a WSN digital forensic readiness prototype.
- Finally, the WSN digital forensic readiness prototype that was implemented, provided feasible results. From the demonstrations we could conclude that the WSN digital forensic readiness model implemented in the prototype was successful in providing a digital forensic readiness layer to an existing WSN.

When all these factors are taken into consideration, it is clear that a feasible way is hereby proposed not only to provide digital forensic readiness to WSNs, but also to detect flooding attacks in WSNs. Although this in itself is already of benefit to the WSN domain, there is still room for improvement.

Future research can be conducted by proposing a better communication protocol for WSNs. If a better communication protocol is found, it can be combined with the WSN digital forensic readiness model. Ideally, this improved communication protocol should in itself be resilient and resistant against flooding attacks. This would allow the WSN digital forensic readiness model to be viable in an environment prone to constant flooding attacks. Currently, if flooding attacks occur at a constant rate, it would be difficult for the oWSN to go out into the field every time to eliminate the flooding mote.

One could also look into improving the flooding detection algorithm. This algorithm could most likely be enriched by implementing an artificial intelligence technique that is currently still unknown.

From this research, several papers have been published for presentation at conferences. The papers which were published include Mouton and Venter (2009, 2011a, 2011b). These papers are included in Appendix B. Also included in Appendix B is Bezuidenhout, Mouton and Venter (2010), a paper based on social engineering and also published during this research. It is, however, not directly related to this research.

Bibliography

- Akyildiz, I.F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) 'Wireless sensor networks: a survey', *Computer Networks*, vol. 38, no. 4, pp. 393-422.
- Bace, R.G. (2000) *Intrusion Detection*, Indianapolis: Macmillan Computer Publishing.
- Baryamureeba, V. and Tushabe, F. (2004) 'The Enhanced Digital Investigation Process Model', Digital Forensics Research Workshop.
- Beebe, N.L. and Clark, J. (2004) 'A Hierarchical, Objectives-Based Framework for the Digital Investigations Process', Digital Forensics Research Workshop, Baltimore.
- Bernstein, D.E. (2000) 'Frye, Frye, Again: The Past, Present, and Future of the General Acceptance Test', *Jurimetrics*, vol. 41, no. 1, Fall, p. 385.
- Bezuidenhout, M., Mouton, F. and Venter, H.S. (2010) 'Social engineering attack detection model: SEADM', Information Security for South Africa, Johannesburg, 1-8.
- Callaway, E., Gorday, P., Hester, L., Gutierrez, J.A., Naeve, M., Heile, B. and Bahl, V. (2002) 'Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks', *Communications Magazine, IEEE*, vol. 40, no. 8, August, pp. 70-77.
- Carrier, B. (2002) *Open source digital forensics tools: The legal argument*, Stake Research Report.
- Carrier, B. and Spafford, E. (2003) 'Getting Physical with the Digital Investigation Process', *International Journal of Digital Evidence*, vol. 2, no. 2, Fall.
- Casey, E. (2002) 'Error, Uncertainty and Loss in Digital Evidence', *International Journal of Digital Evidence*, vol. 1, no. 2, Summer.
- Chen, J., Jiang, M. and Liu, Y. (2005) 'Wireless LAN security and IEEE 802.11i', *Wireless Communications, IEEE*, vol. 12, no. 1, February, pp. 27-36.
- Cheswick, W.R., Bellovin, S.M. and Rubin, A.D. (2003) *Firewalls and Internet Security: Repelling the Wily Hacker*, Boston: Addison-Wesley Longman Publishing Company Incorporated.
- Chinrungrueng, J., Sunantachaikul, U. and Triamlumlerd, S. (2007) 'Smart Parking: An Application of Optical Wireless Sensor Network', IEEE/IPSJ International Symposium on Applications and the Internet Workshops, Hiroshima, 66.
- Chong, C. and Kumar, S.P. (2003) 'Sensor networks: evolution, opportunities, and challenges', *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247-1256.

- Chun-Hsien, W., Kuo-Chuan, L. and Yeh-Ching, C. (2007) 'A Delaunay Triangulation based method for wireless sensor network deployment', *Computer Communications*, vol. 30, October, pp. 2744-2752.
- Crossbow Technology Inc (2005) *Radio, RF Concepts, and TOS radio stack.*, San Jose: Crossbow Technology Inc.
- Crossbow Technology Inc (2005) *TPR2400/2420 Quick Start Guide*, Revision A edition, San Jose: Crossbow Technology Inc.
- Crossbow Technology Inc (2007) *Imote2 Hardware Reference Manual*, Revision A edition, San Jose: Crossbow Technology Inc.
- Crossbow Technology Inc (2007) *Imote2.Builder Kit Manual*, Revision A edition, San Jose: Crossbow Technology Inc.
- Crow, B.P., Widjaja, I., Kim, L.G. and Sakai, P.T. (1997) 'IEEE 802.11 Wireless Local Area Networks', *IEEE Communications Magazine*, vol. 35, no. 9, pp. 116-126.
- Deal, R. (2004) *Cisco Router Firewall Security*, Cisco Press.
- Debar, H., Dacier, M. and Wespi, A. (2000) 'A revised taxonomy for intrusion-detection systems', *Annals of Telecommunications*, vol. 55, no. 7, pp. 361-378.
- Denning, D.E. and Smid, M. (1994) 'Key escrowing today', *IEEE Communications Magazine*, vol. 32, no. 9, pp. 58-68.
- Doolin, D. and Sitar, N. (2005) 'Wireless sensors for wildfire monitoring', In Proceedings of SPIE Symposium on Smart Structures & Materials, San Diego.
- Doufexi, A., Armour, S., Butler, M., Nix, A., Bull, D., McGeehan, J. and Karlsson, P. (2002) 'A comparison of the HIPERLAN/2 and IEEE 802.11a wireless LAN standards', *IEEE Communications Magazine*, vol. 40, no. 5, May, pp. 172-180.
- Dunkels, A., Osterlind, F. and Zhitao, H. (2007) 'An adaptive communication architecture for wireless sensor networks', Proceedings of the 5th international conference on Embedded networked sensor systems, Sydney, Australia, 335-349.
- Elson, J. and Estrin, D. (2001) 'Time synchronization for wireless sensor networks', Proceedings of the 15th International Symposium on Parallel and Distributed Processing, 1965-1970.

- Estrin, D., Girod, L., Pottie, G. and Srivastava, M. (2001) 'Instrumenting the world with wireless sensor networks', Proceedings of the 2001 IEEE International Conference on Acoustics, Speech and Signal Processing, 2033-2036.
- Ford, B., Srisuresh, P. and Kegel, D. (2005) 'Peer-to-Peer Communication Across Network Address Translators', In Proceedings of the 2005 USENIX Annual Technical Conference, Anaheim, CA, USA.
- Garber, L. (2000) 'Denial-of-service attacks rip the internet', *Computer*, vol. 33, no. 4, April, pp. 12-17.
- Gehlen, G., Aijaz, F. and Walke, B. (2006) 'Mobile Web Service Communication Over UDP', Vehicular Technology Conference, 1-5.
- Gianelli, P.C. and Imwinkelried, E.J. (1999) *Scientific evidence*, Charlottesville, VA: LEXIS Law Pub.
- Gouda, M.G. and Liu, A.X. (2005) 'A Model of Stateful Firewalls and Its Properties', International Conference on Dependable Systems and Networks, Los Alamitos, 128-137.
- Grobler, C.P., Louwrens, C.P. and von Solms, S.H. (2010) 'A Framework to Guide the Implementation of Proactive Digital Forensics in Organisations', International Conference on Availability, Reliability and Security, Los Alamitos, CA, USA, 677-682.
- Guizani, M. and Raju, A. (2005) 'Wireless Networks and Communications Security', in Xiao, Y., Li, J. and Pan, Y. (ed.) *Security and Routing in Wireless Networks*, 3rd edition, New York: Nova Science Publishers.
- Guo, F. and Chiueh, T. (2006) *Sequence Number-Based MAC Address Spoof Detection*, Heidelberg: Springer Berlin.
- Harris, B. and Hunt, R. (1999) 'TCP/IP security threats and attack methods', *Computer Communications*, vol. 22, no. 10, pp. 885-897.
- Hartley, R.I. and Sturm, P. (1997) 'Triangulation', *Computer Vision and Image Understanding*, vol. 68, no. 2, November, pp. 146-157.
- Heinzelman, W.R., Kulik, J. and Balakrishnan, H. (1999) 'Adaptive protocols for information dissemination in wireless sensor networks', In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, 174-185.
- Hong, S., Hong, D., Ko, Y., Chang, D., Lee, W. and Lee, S. (2004) 'Differential Cryptanalysis of TEA and XTEA', *Information Security and Cryptology*, vol. 2971, pp. 402-417.

- Jonakait, R.N. (1993) 'The Meaning of Daubert and What that Means for Forensic Science', *Cardozo Law Review*, vol. 15, no. 7, October, p. 2103.
- Kahn, J.M., Katz, R.H. and Pister, K.S. (1999) 'Next century challenges: mobile networking for "Smart Dust"', In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, New York, 271-278.
- Karig, D. and Lee, R. (2001) 'Remote Denial of Service Attacks and Countermeasures', Department of Electrical Engineering Technical Report CE-L2001-002, Princeton University.
- Karlof, C. and Wagner, D. (2003) 'Secure routing in wireless sensor networks: attacks and countermeasures', *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293-315.
- Khoussainov, R. and Patel, A. (2000) 'LAN security: problems and solutions for Ethernet networks', *Computers Standards & Interfaces*, vol. 22, no. 3, August.
- Kim, T.H., Choi, Y.-S., Kim, J. and Hong, S.J. (2008) 'Annulling SYN Flooding Attacks with Whitelist', International Conference on Advanced Information Networking and Applications Workshops, Los Alamitos, 371-376.
- Komori, T. and Saito, T. (2004) 'A secure wireless LAN system retaining privacy', 18th International Conference on Advanced Information Networking and Applications, Kanagawa, 370-375.
- Lau, F., Rubin, S.H., Smith, M.H. and Trajovic, L. (2000) 'Distributed denial of service attacks', In IEEE International Conference on Systems, Man, and Cybernetics, Nashville, 2275-2280.
- Leigland, R. and Krings, A.W. (2004) 'A Formalization of Digital Forensics', *International Journal of Digital Evidence*, vol. 3, no. 2.
- Lemon, J. (2002) 'Resisting SYN Flooding DoS Attacks with a SYN Cache', Proceedings of USENIX BSDCon'2002.
- Lewis, F.L. (2005) *Smart Environments*, John Wiley & Sons, Inc.
- Lu, C., Blum, B.M., Abdelzaher, T.F., Stankovic, J.A. and He, T. (2002) 'RAP: a real-time communication architecture for large-scale wireless sensor networks', Proceedings of the 8th IEEE International Workshop on Real-Time and Embedded Technology and Applications Symposium, 55-66.
- Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R. and Anderson, J. (2002) 'Wireless sensor networks for habitat monitoring', In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, New York, 88-97.

- Maroti, M., Kusy, B., Simon, G. and Ledeczi, A. (2004) 'The flooding time synchronization protocol', Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, MD, USA, 39-49.
- McHugh, J., Christie, A. and Allen, J. (2000) 'Defending yourself: the role of intrusion detection systems', *Software, IEEE*, vol. 17, no. 5, October, pp. 42-51.
- Meyers, M. and Rogers, M. (2004) 'Computer Forensics: The Need for Standardization and Certification', *International Journal of Digital Evidence*, vol. 3, no. 2.
- Mills, D.L. (1991) 'Internet time synchronization: the network time protocol', *IEEE Transactions on Communications*, vol. 39, no. 10, October, pp. 1482 -1493.
- Mirkovic, J., Dietrich, S., D, D. and Reither, P. (2004) *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)*, New Jersey: Prentice Hall PTR.
- Mirkovic, J. and Reihher, P. (2004) 'A taxonomy of DDoS attack and DDoS defense mechanisms', *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, April, pp. 39-53.
- Mocas, S. (2004) 'Building Theoretical Underpinnings for Digital Forensics Research', *Digital Investigation*, vol. 1, pp. 61-68.
- Mogul, J. and Postel, J. (1985) *Internet Standard Subnetting Procedure*, Internet RFC 950.
- Moore, D., Shannon, C., Brown, D.J., Voelker, G.M. and Savage, S. (2006) 'Inferring Internet denial-of-service activity', *ACM Transactions on Computer Systems*, vol. 24, no. 2, May, pp. 115-139.
- Moteiv Corporation (2004) *Telos. Rev B (Low Power Wireless Sensor Module)*, El Cerrito: Moteiv Corporation.
- Mouton, F. and Venter, H.S. (2009) 'A Secure Communication Protocol for Wireless Sensor Networks', Proceedings of the Annual Security Conference "Security Assurance and Privacy: organizational challenges", Las Vegas.
- Mouton, F. and Venter, H.S. (2011) 'Requirements for Wireless Sensor Networks in Order to Achieve Digital Forensic Readiness', Proceedings of the 6th International Workshop on Digital Forensics & Incident Analysis, London, 108-121.
- Mouton, F. and Venter, H.S. (2011) 'A prototype for achieving digital forensic readiness on wireless sensor networks', Proceedings of the 10th IEEE Africon, Zambia.

- Mukkamala, S. and Sung, A.H. (2004) 'Computational Intelligent Techniques for Detecting Denial of Service Attacks', in Orchard, B., Yang, C. and Ali, M. *Innovations in Applied Artificial Intelligence*, Heidelberg: Springer Berlin.
- Ni, S.I., Tseng, I.C. and Sheu, J.P. (2001) 'Efficient Broadcasting in a Mobile Ad Hoc Network', Proceedings of the IEEE International Conference on Distributed Computing and Systems, 16-19.
- Oppliger, R. and Rytz, R. (2003) 'Digital Evidence: Dream and Reality', *IEEE Security & Privacy*, vol. 1, pp. 44-48.
- Orecchia, L., Panconesi, A., Petrioli, C. and Vitaletti, A. (2004) 'Localized techniques for broadcasting in wireless sensor networks', In Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing, New York, 41-51.
- Oxford English Dictionary*, 2nd edition (1989), Oxford: Clarendon Press.
- Palmer, G.L. (2002) 'A Road Map for Digital Forensics Research', First Digital Forensics Research Workshop, 1-48.
- Perrig, A., Stankovic, J. and Wagner, D. (2004) 'Security in wireless sensor networks', *Communications of the ACM*, vol. 47, pp. 53-57.
- Perrig, A., Szewczyk, R., Wen, R., Culler, D. and Tygar, J. (2001) 'SPINS: Security Protocols for Sensor Networks', In Seventh Annual ACM International Conference on Mobile Computing and Networks, Rome, 521-234.
- Pfleeger, C.P. and Pfleeger, S.L. (2006) *Security in Computing (4th Edition)*, New Jersey: Prentice Hall PTR.
- Polastre, J., Hill, J. and Culler, D. (2004) 'Versatile low power media access for wireless sensor networks', Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, 95-107.
- Pollitt, M.M. (2007) 'An Ad Hoc Review of Digital Forensic Models', IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering, Los Alamitos, CA, USA, 43-54.
- Reith, M., Carr, C. and Gunsch, G. (2002) 'An Examination of Digital Forensic Models', *International Journal of Digital Evidence*, vol. 1, no. 3, Fall.

- Rogers, M.K. and Seigfried, K. (2004) 'The future of computer forensics: a needs analysis survey', *Computers & Security*, vol. 23, pp. 12-16.
- Rowlingson, R. (2004) 'A Ten Step Process for Forensic Readiness', *International Journal of Digital Evidence*, vol. 2, no. 3.
- Santi, P. (2005) 'Topology control in wireless ad hoc and sensor networks', *ACM Computer Survey*, vol. 37, no. 2, pp. 164-194.
- Savarese, C., Rabaey, J.M. and Beutel, J. (2001) 'Location in distributed ad-hoc wireless sensor networks', In The Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2037-2040.
- Schatz, B., Mohay, G. and Clark, A. (2006) 'A correlation method for establishing provenance of time stamps in digital evidence', Proceedings of the 6th Annual Digital Forensics Research Workshop, 98-107.
- Schneier, B. and Kelsey, J. (1999) 'Secure audit logs to support computer forensics', *ACM Transactions Information System Security*, vol. 2, no. 2, May, pp. 159-176.
- Shnayder, V., Hempstead, M., Chen, B., Allen, G.W. and Welsh, M. (2004) 'Simulating the power consumption of large-scale sensor network applications', In Proceedings of the 2nd international Conference on Embedded Network Sensor Systems, Baltimore, 188-200.
- Slijepcevic, S. and Potkonjak, M. (2001) 'Power efficient organization of wireless sensor networks', In IEEE International Conference on Communications, 472-476.
- Slijepcevic, S., Potkonjak, M., Tsiatsis, V., Zimbeck, S. and Srivastava, M.B. (2002) 'On communication security in wireless ad-hoc sensor networks', *Enabling Technologies: Infrastructure for Collaborative Enterprises*, 139-144.
- Sohrabi, K., Gao, J., Ailawadhi, V. and Pottie, G.J. (2000) 'Protocols for self-organization of a wireless sensor network', *Personal Communications, IEEE Wireless communications*, vol. 7, no. 5, October, pp. 16-27.
- Spurgeon, C.E. (2000) *Ethernet: the Definitive Guide*, Austin: O'Reilly & Associates, Inc.
- Su, W. and Akyildiz, I.F. (2005) 'Time-diffusion synchronization protocol for wireless sensor networks', *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 384-397.
- Sundararaman, B., Buy, U. and Kshemkalyani, A.D. (2005) 'Clock synchronization for wireless sensor networks: a survey', *Ad Hoc Networks*, vol. 3, no. 3, pp. 281-323.

- Sun, K., Ning, P. and Wang, C. (2006) 'TinySerSync: secure and resilient time synchronisation in wireless sensor networks', Proceedings of the 13th ACM conference on Computer and communications security, Alexandria, 264-277.
- Tan, J. (2001) *Forensic Readiness*, Technical Report edition, Cambridge: @Stake.
- Templeton, S.J. and Levitt, K. (2000) 'A requires/provides model for computer attacks', Proceedings of the 2000 workshop on New security paradigms, Ballycotton, 31-38.
- Thai, T. and Lam, H. (2003). *NET Framework Essentials*, 3rd edition, Sebastopol: O'Reilly.
- Thompson, D. and Miles, R. (2007) *Embedded programming with the microsoft.net micro framework*, Redmond: Microsoft Press.
- Tseng, Y., Ni, S. and Shih, E. (2003) 'Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network', *IEEE Transactions on Computers*, vol. 52, no. 5, pp. 545-557.
- Venter, H.S. (2003) *A model for vulnerability forecasting*, University of Johannesburg: PhD Thesis.
- Vibration Engineering Research Section (2009) *Imote2 discussion forum*, 2009 December, [Online], Available: HYPERLINK "http://vibration.shef.ac.uk/imote2_forum/"
http://vibration.shef.ac.uk/imote2_forum/ .
- Walrand, J. (1998) *Communication Networks: A First Course*, New York: Mc-Graw-Hill.
- Wander, A.S., Gura, N., Eberle, H., Gupta, V. and Shantz, S.C. (2005) 'Energy analysis of public-key cryptography for wireless sensor networks', Third IEEE International Conference on Pervasive Computing and Communications, 324-328.
- Wang, K. and Stolfo, S.J. (2004) 'Anomalous Payload-Based Network Intrusion Detection', in Jonsson, E., Valdes, A. and Almgren, M. *Recent Advances in Intrusion Detection*, Heidelberg: Springer Berlin.
- Wheeler, A. (2007) 'Commercial Applications of Wireless Sensor Networks Using ZigBee', *Communications Magazine, IEEE*, vol. 45, no. 4, April, pp. 70-77.
- Wood, A.D. and Stankovic, J.A. (2002) 'Denial of service in sensor networks', *Computer*, vol. 35, no. 10, October, pp. 54-62.
- Wu, J. and Stojmenovic, I. (2004) 'Ad hoc networks', *Computer*, vol. 37, no. 2, February, pp. 29-31.

- Xiang-Yang, L., Calinescu, G., Peng-Jun, W. and Yu, W. (2003) 'Localized Delaunay triangulation with application in ad hoc wireless networks', *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 10, October, pp. 1035-1047.
- Xylomenos, G. and Polyzos, G. (1999) 'TCP and UDP Performance over a Wireless LAN', In Proceedings of the IEEE INFOCOM.
- Xylomenos, G., Polyzos, G., Mahonen, P. and Saaranen, M. (2001) 'TCP performance issues over wireless links', *IEEE Communications Magazine*, vol. 39, no. 4, pp. 52-58.
- Yasinsac, A., Erbacher, R.F., Marks, D.G., Politt, M.M. and Sommer, P.M. (2003) 'Computer Forensics Education', *IEEE Security & Privacy*, pp. 15-23.
- Ye, W., Heidemann, J. and Estrin, D. (2002) 'An energy-efficient MAC Protocol for wireless sensor networks', Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communication Societies, 1567-1576.
- Zhang, L., Zhang, D. and Wang, L. (2010) 'Live digital forensics in a virtual machine', International Conference on Computer Application and System Modeling (ICCASM), Taiyuan, China, 328-332.
- Zhao, F., Shin, J. and Reich, J. (2002) 'Information-driven dynamic sensor collaboration for tracking applications', *IEEE Signal Processing Magazine*, vol. 19, pp. 61-72.
- Zhu, S., Setia, S. and Jajodia, S. (2006) 'LEAP+: Efficient security mechanisms for large-scale distributed sensor networks', *ACM Transactions on Sensor Networks*, vol. 2, no. 4, pp. 500-528.

Appendix A: Source Code

A.1 fWSN Field Mote

```

using System;
using Microsoft.SPOT;
using Crossbow.platform.imote2;
using Crossbow.lib.utils;
using Crossbow.lib.utils.tos;
using Crossbow.radio.cc2420;

namespace Crossbow.app
{
    public class ForensicMote
    {
        public static void Main()
        {
            int BLACK = 0;
            int RED = 1;
            int BLUE = 2;
            int PURPLE = 3;
            int GREEN = 4;
            int YELLOW = 5;
            int TEAL = 6;
            int WHITE = 7;
            Leds _leds = new Leds();

            const ushort TOS_UART_ADDR = 0x7E;
            const ushort _rfChannel = (ushort)RadioChannel.Ch11; // channel 26
            const ushort _rfPower = (ushort)RadioPower.M10DBM; // -10dbm
            const int _default_nodeid = 0x01;
            const int _amType = 50;

            int _nodeid = UniqueID.SerialID;

            _nodeid = 0xCAFF; // set to default if there is not UID in the flash

            SerialDump.print(_nodeid + " Node ID");
            byte moteID = 202;

            SerialDump.print("ForensicMote App started"); //Print out the application identifier

            TOSRadio _radio = new TOSRadio();
            _radio.SetRadioOption(RadioOption.Frequency, _rfChannel);
            _radio.SetRadioOption(RadioOption.RFPower, _rfPower);
            _radio.SetRadioOption(RadioOption.PANAddress, (ushort)_nodeid);
            _radio.SetRadioOption(RadioOption.LocalAddress, (ushort)_nodeid);

            byte[] convertedBuf;
            TosMsg recvMsg;

            _leds.setRGB(YELLOW);

            while (true)
            {
                if (null != (recvMsg = _radio.PromiscuousListening(20)))
                {
                    _leds.setRGB(YELLOW);

                    // PC side only recognizes mica2 TOS message format

                    recvMsg.addr = TOS_UART_ADDR; // put in the UART address
                    convertedBuf = recvMsg.ConvertToMica2Msg();
                    // SerialDump.print(convertedBuf);
                    if (convertedBuf[5] != 100 && convertedBuf[6] != 100 && convertedBuf[7] != 100)
                    {
                        byte[] packet = new byte[64];

                        packet[0] = 100;
                        packet[1] = 100;
                        packet[2] = 100;
                        packet[3] = moteID;
                        for (int i = 4; i < 64 && i < convertedBuf.Length; i++)
                        {

```

```

        packet[i] = convertedBuf[i - 4];
    }
    SerialDump.print(packet);           //send it to the PC

    SerialDump.print("Packet being sent to Forensic Base");
    SerialDump.print(BufferToString(packet));
    SerialDump.print(BufferToString(convertedBuf));
    int randomNumber = Microsoft.SPOT.Math.Random(275) + 250;
    System.Threading.Thread.Sleep(randomNumber);
    if (_radio != null)
    {

        // send it to the radio

        _radio.SetRadioOption(RadioOption.LocalAddress, (ushort)_nodeid);
        _radio.SetRadioOption(RadioOption.Frequency, (ushort)RadioChannel.Ch12);
        _radio.amType = _amType;

        _radio.Send((ushort)0xFFFF, (ushort)0xFFFF, packet);
        _radio.SetRadioOption(RadioOption.Frequency, (ushort)RadioChannel.Ch11);

        SerialDump.print("Transmitted Packet to Forensic Base");
    }
    _radio.SetRadioOption(RadioOption.Frequency, (ushort)RadioChannel.Ch11);
    _leds.setRGB(YELLOW);
}
else
{
    SerialDump.print("Message from nearby Forensic Motes");
}
}
}

public static byte[] StrToByteArray(string str)
{
    return System.Text.UTF8Encoding.UTF8.GetBytes(str);
}

public static string ByteArrayToStr(byte[] dBytes)
{
    char[] temp = System.Text.UTF8Encoding.UTF8.GetChars(dBytes);
    return new string(temp);
}

private static string ByteToHex(byte b)
{
    const string hex = "0123456789ABCDEF";
    int lowNibble = b & 0x0F;
    int highNibble = (b & 0xF0) >> 4;
    string s = new string(new char[] { hex[highNibble], hex[lowNibble] });
    return s;
}

private static string BufferToString(byte[] buffer)
{
    if (buffer == null)
        throw new ArgumentNullException("buffer");
    string s = string.Empty;
    for (int i = 0; i < buffer.Length; i++)
    {
        s += ByteToHex(buffer[i]) + " ";
        if (i > 0 && i % 16 == 0)
            s += "\n";
    }
    return s;
}
}
}
}

```

A.2 fWSN Base Station

```

using System;
using Microsoft.SPOT;
using Crossbow.platform.imote2;
using Crossbow.lib.utils;
using Crossbow.lib.utils.tos;
using Crossbow.radio.cc2420;

namespace Crossbow.app
{
    public class ForensicBase
    {
        public static void Main()
        {
            const ushort TOS_UART_ADDR = 0x7E;
            const ushort _rfChannel = (ushort)RadioChannel.Ch11; // channel 11
            const ushort _rfPower = (ushort)RadioPower.M1DBM; // -10dbm

            SerialDump.print("ForensicBase App started"); //Print out the application identifier

            Leds _leds = new Leds();

            TOSRadio _radio = new TOSRadio(_rfChannel, _rfPower, 0xFF01, 0xFF01);
            TosMsg recvMsg;
            byte[] convertedBuf;

            _leds.setRGB(0x00FFFF);

            while (true)
            {
                if (null != (recvMsg = _radio.PromiscuousListening(10)))
                {
                    // PC side only recognizes mica2 TOS message format

                    recvMsg.addr = TOS_UART_ADDR; // put in the UART address
                    convertedBuf = recvMsg.ConvertToMica2Msg();
                    if (convertedBuf[5] == 100 && convertedBuf[6] == 100 && convertedBuf[7] == 100)
                    {

                        _leds.setRGB(0xFFFF00);

                        SerialDump.print(convertedBuf); //send it to the PC

                        _leds.setRGB(0x00FFFF);
                    }
                }
            }

            public static byte[] StrToByteArray(string str)
            {
                return System.Text.UTF8Encoding.UTF8.GetBytes(str);
            }

            public static string ByteArrayToStr(byte[] dBytes)
            {
                char[] temp = System.Text.UTF8Encoding.UTF8.GetChars(dBytes);
                return new string(temp);
            }

            private static string ByteToHex(byte b)
            {
                const string hex = "0123456789ABCDEF";
                int lowNibble = b & 0x0F;
                int highNibble = (b & 0xF0) >> 4;
                string s = new string(new char[] { hex[highNibble], hex[lowNibble] });
                return s;
            }

            private static string BufferToString(byte[] buffer)
            {
                if (buffer == null)
                    throw new ArgumentNullException("buffer");
                string s = string.Empty;

```

```
for (int i = 0; i < buffer.Length; i++)  
{  
    s += ByteToHex(buffer[i]) + " ";  
    if (i > 0 && i % 16 == 0)  
        s += "\n";  
}  
return s;  
}  
}
```

A.3 Forensic Packet Logging Software

```

using System;
using System.IO;
using System.Text;
using System.Collections;
using System.Reflection;
using System.Threading;
using System.Diagnostics;

using _DBG = Microsoft.SPOT.Debugger;
using _WP = Microsoft.SPOT.Debugger.WireProtocol;

namespace Microsoft.SPOT.Tools
{
    class SerialDump
    {
        _DBG.Engine m_eng;

        bool m_fTimestamp = true;
        bool m_fUsb = true;
        bool m_fNewLine = true;
        FileStream m_output;
        int counter;
        DateTime started;

        //--//

        SerialDump(string[] args)
        {
            _DBG.PortDefinition pd = null;
            string port = null;
            uint baudrate = 0;
            counter = 1;

            started = DateTime.Now;
            m_fUsb = true;

            if (m_fUsb)
            {
                _DBG.PortDefinition[] ports = _DBG.AsyncUsbStream.EnumeratePorts();

                if (port == null)
                {
                    if (ports.Length == 0)
                    {
                        System.Console.WriteLine("No Usb SPOT device is present");

                        throw new ApplicationException();
                    }
                    else if (ports.Length == 1)
                    {
                        pd = ports[0];
                    }
                    else
                    {
                        System.Console.WriteLine("More than one USB SPOT device is present");
                        System.Console.WriteLine("To dump data from a specific device, choose your device from the list below and execute:");
                        System.Console.WriteLine(" serialdump -usb <device> ");
                        System.Console.WriteLine("");

                        //
                        // More than one usb device attached; dump list so user can choose
                        //
                        for (int i = 0; i < ports.Length; ++i)
                        {
                            System.Console.WriteLine("Device " + i + ": " + ports[i].DisplayName);
                        }

                        throw new ApplicationException();
                    }
                }
            }
            else
            {
                foreach (_DBG.PortDefinition pd2 in ports)
                {

```

```

        if (port.Equals(pd2.DisplayName))
        {
            pd = pd2;
            break;
        }
    }
}

if (pd == null)
{
    if (port == null) port = "COM1";
    if (baudrate == 0) baudrate = 115200;

    pd = _DBG.PortDefinition.CreateInstanceForSerial(port, port, baudrate);
}

m_eng = new _DBG.Engine(pd);
}

void Run()
{
    m_eng.Silent = true;

    m_eng.Start();

    m_eng.OnNoise += new _DBG.NoiseEventHandler(OnNoise);
    m_eng.OnMessage += new _DBG.MessageEventHandler(OnMessage);

    Console.WriteLine("##### PRESS RETURN TO EXIT #####");
    Console.ReadLine();

    m_eng.Stop();
    m_eng = null;
}

void OnNoise(byte[] buf, int offset, int count)
{
    HandleOutput(buf, offset, count);
}

void OnMessage(_WP.IncomingMessage msg, string text)
{
    byte[] buf = Encoding.ASCII.GetBytes(text);

    HandleOutput(buf, 0, buf.Length);
}

void HandleOutput(byte[] buf, int offset, int count)
{
    while (count-- > 0)
    {
        if (m_fTimestamp)
        {
            if (m_fNewLine)
            {
                DateTime now = DateTime.Now;
                if (now.Minute != started.Minute)
                {
                    counter++;

                    m_output.Flush();
                    m_output.Close();
                    started = DateTime.Now;
                }
                now = DateTime.Now;
                if (!File.Exists("C:\\wsnlog\\" + String.Format("[{0:dd-MM-yyyy}]",
                    DateTime.Now) + "_log_" + counter + ".txt"))
            {

```



```
        m_output = new FileStream("C:\\wsnlog\\" + String.Format("{0:dd-MM-
            yyyy}]", DateTime.Now) + "_log_" + counter + ".txt",
            FileMode.Create);
    }

    HandleOutput(String.Format("[{0:dd/MM/yyyy HH:mm:ss:fff zzz}] ",
        DateTime.Now));
    }
}

char c = (char)buf[offset++];

HandleOutput(new String(c, 1));

m_fNewLine = (c == '\n');
}
}

void HandleOutput(string text)
{
    Console.Write(text);

    byte[] buf = Encoding.ASCII.GetBytes(text);

    if (m_output != null) m_output.Write(buf, 0, buf.Length);
}

//--//

static void Main(string[] args)
{
    try
    {
        SerialDump o = new SerialDump(args);

        o.Run();
    }
    catch (ApplicationException)
    {
    }
    catch (Exception e)
    {
        Console.WriteLine("{0}", e.ToString());
    }
}
}
}
```

A.4 Forensic Packet Analysis Software

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsApplication1
{
    public partial class Form1 : Form
    {
        protected DataGridView dataGridView1 = null;
        DataTable tdt = new DataTable();
        int logCount;

        Timer Clock;

        int flooding = 0;
        int lineNumber = 1;

        public Form1()
        {
            InitializeComponent();
        }

        DataColumn getNewColumn(string ColName, string ColType)
        {
            //This function simply creates a new column to be used in the DataTable
            DataColumn dc = new DataColumn();
            dc.ColumnName = ColName;
            dc.DataType = System.Type.GetType(ColType);
            return dc;
        }

        void CreateNewRow(string [,] list, int count)
        {
            //Create a DataRow based on the DataTable

            for(int i = 0; i < count; i++)
            {
                if (list[i, 63].CompareTo("del") != 0)
                {
                    DataRow ARow = tdt.NewRow();
                    ARow["Line"] = lineNumber++;
                    ARow["Date"] = list[i, 0];
                    ARow["Forensic MoteID"] = list[i, 1];
                    for (int k = 1; k < 61; k++)
                    {
                        if (k == 3)
                        {
                            ARow["Originating MoteID"] = list[i, k + 1];
                        }
                        else if (k == 8)
                        {
                            ARow["Light Value"] = list[i, k + 1];
                        }
                        else if (k == 9)
                        {
                            ARow["Humidity Value"] = list[i, k + 1];
                        }
                        else if (k == 10)
                        {
                            ARow["Temperature Value"] = list[i, k + 1];
                        }
                        else
                        {
                            ARow["Byte " + k] = list[i, k + 1];
                        }
                    }
                    ARow["Seen By"] = list[i, 62];

                    //Now that we have all the values in the DataRow, we add the row to
                    //the DataTable
                }
            }
        }
    }
}

```

```

        tdt.Rows.Add(ARow);
    }
}

private void Form1_Load(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Maximized;

    /*
    Add Columns to the datatable
    We're not using all these columns, but at first I thought they may
    be useful, so I left them in and simply hide them lower down
    */
    tdt.Columns.Add(getNewColumn("Line", "System.String"));
    tdt.Columns.Add(getNewColumn("Date", "System.String"));
    tdt.PrimaryKey = new DataColumn[] { tdt.Columns["ID"] };
    tdt.Columns.Add(getNewColumn("Seen By", "System.String"));
    tdt.Columns.Add(getNewColumn("Forensic MoteID", "System.String"));
    for (int i = 1; i < 61; i++)
    {
        if (i == 3)
        {
            tdt.Columns.Add(getNewColumn("Originating MoteID", "System.String"));
        }
        else if (i == 8)
        {
            tdt.Columns.Add(getNewColumn("Light Value", "System.String"));
        }
        else if (i == 9)
        {
            tdt.Columns.Add(getNewColumn("Humidity Value", "System.String"));
        }
        else if (i == 10)
        {
            tdt.Columns.Add(getNewColumn("Temperature Value", "System.String"));
        }
        else
        {
            tdt.Columns.Add(getNewColumn("Byte " + i, "System.String"));
        }
    }

    logCount = 1;

    string line;
    //Open a text file for reading
    System.IO.StreamReader file = new System.IO.StreamReader(@"C:\\wsnlog\\" +
        String.Format("[{0:dd-MM-yyyy}]", DateTime.Now) + "_log_" + logCount + ".txt");

    // Read the file and add it to the datagridview, if google related

    string[] initial = new string[100000];
    int count = 0;

    //MessageBox.Show("The calculations are complete", "My Application");

    while ((line = file.ReadLine()) != null)
    {
        if (line.Contains("100 100 100"))
        {
            initial[count] = line;
            count++;
        }
    }

    string[,] list = new string[count, 64];
    for (int index = 0; index < count; index++)
    {
        line = initial[index];
        int ColStart = 0;

```

```

int ColEnd = 0;
string ColValue = "";
//Get the date from the line parameter
list[index, 0] = line.Substring(1, 23);

ColStart = 34;
for (int i = 0; i < 8; i++)
{
    ColEnd = line.IndexOf(" ", ColStart);
    ColValue = line.Substring(ColStart, ColEnd - ColStart);

    ColStart = ColEnd + 1;
}

ColEnd = line.IndexOf(" ", ColStart);

//Forensic MoteID
list[index, 1] = line.Substring(ColStart, ColEnd - ColStart);
list[index, 62] = list[index, 1];
ColStart = ColEnd + 1;

//Move one character on and repeat the process for the remaining columns
for (int i = 1; i < 61; i++)
{
    ColEnd = line.IndexOf(" ", ColStart);
    ColValue = line.Substring(ColStart, ColEnd - ColStart);
    list[index, i + 1] = ColValue;
    ColStart = ColEnd + 1;
}
list[index, 63] = "a";
}
/* Array INDEXES
* 0 = Date
* 1 = Which Forensic Mote Received it
* 2 - 61 = bytes
* 62 = also seen by
*/

for (int i = 0; i < count; i++)
{
    if (list[i, 63].CompareTo("del") != 0)
    {
        for (int j = 0; j < count; j++)
        {
            if (i != j)
            {
                bool samePacket = true;
                for (int k = 2; k < 62; k++)
                {
                    if (list[i, k].CompareTo(list[j, k]) != 0 || list[i,
                        62].Contains(list[j, 1]))
                    {
                        samePacket = false;
                    }
                    if (!samePacket)
                    {
                        break;
                    }
                }
                if (samePacket)
                {
                    list[i, 62] = list[i, 62] + ", " + list[j, 1];
                    list[j, 63] = "del";
                }
            }
        }
    }
}

CreateNewRow(list, count);

// MessageBox.Show(count + " " + list.Length);

file.Close();

```

```

/*
Create the DataGridView programatically
*/
dataGridView1 = new DataGridView();
this.Controls.Add(dataGridView1);
dataGridView1.ColumnCount = 1;
DataGridViewCellStyle style = dataGridView1.ColumnHeadersDefaultCellStyle;
style.BackColor = Color.BlueViolet;
style.ForeColor = Color.Black;
style.Font = new Font(dataGridView1.Font, FontStyle.Bold);
// dataGridView1.EditMode = DataGridViewEditMode.EditOnEnter;
dataGridView1.Name = "dataGridView1";
dataGridView1.Location = new Point(10, 10);
dataGridView1.Size = new Size(this.Width - 20, this.Height - 200);
dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;
dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
dataGridView1.ColumnHeadersBorderStyle = DataGridViewHeaderBorderStyle.Raised;
dataGridView1.ColumnHeadersHeight = 54;
dataGridView1.CellBorderStyle = DataGridViewCellBorderStyle.Single;
dataGridView1.GridColor = SystemColors.ActiveBorder;
dataGridView1.RowHeadersVisible = false;
dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
dataGridView1.MultiSelect = false;
dataGridView1.BackgroundColor = Color.Honeydew;
dataGridView1.Dock = DockStyle.Fill;
dataGridView1.AutoSize = true;

//Assign the DataTable which we created earlier to the DataGridView
dataGridView1.DataSource = tdt;

for (int i = 0; i < dataGridView1.RowCount - 1; i++)
{
    String CNumColour = dataGridView1.Rows[i].Cells["Seen By"].Value.ToString();
    if (CNumColour.Contains(","))
    {
        dataGridView1.Rows[i].Cells[1].Style.BackColor = Color.Aqua;
    }
}

//Hide some columns we're not interested in
for (int i = 0; i < 1; i++)
{
    dataGridView1.Columns[i].Visible = false;
}

dataGridView1.Columns[13].Visible = false;
//Resize columns for best fit
// dataGridView1.AutoSizeColumns
//(DataGridViewAutoSizeColumnCriteria.HeaderAndDisplayedRows);

Clock = new Timer();
Clock.Interval = 500;
Clock.Start();
Clock.Tick += new EventHandler(Timer_Tick);
flooding = count;
}

public void Timer_Tick(object sender, EventArgs eArgs)
{
    if (sender == Clock)
    {
        logCount++;
        if (logCount < 8)
        {
            string line;
            //Open a text file for reading
            System.IO.StreamReader file = new System.IO.StreamReader(@"C:\wsnlog\" +
                String.Format("{0:dd-MM-yyyy}", DateTime.Now) + "_log_" + logCount +
                ".txt");

            // System.IO.StreamReader aheadFile = new System.IO.StreamReader(@"output.txt");
            // string nextLine = aheadFile.ReadLine();

```

```
// Read the file and add it to the datagridview, if google related

string[] initial = new string[100000];
int count = 0;

while ((line = file.ReadLine()) != null)
{
    if (line.Contains("100 100 100"))
    {
        initial[count] = line;
        count++;
    }
}

string[,] list = new string[count, 64];
for (int index = 0; index < count; index++)
{

    line = initial[index];
    int ColStart = 0;
    int ColEnd = 0;
    string ColValue = "";
    //Get the date from the line parameter
    list[index, 0] = line.Substring(1, 23);

    ColStart = 34;
    for (int i = 0; i < 8; i++)
    {

        ColEnd = line.IndexOf(" ", ColStart);
        ColValue = line.Substring(ColStart, ColEnd - ColStart);

        ColStart = ColEnd + 1;
    }

    ColEnd = line.IndexOf(" ", ColStart);

    //Forensic MoteID
    list[index, 1] = line.Substring(ColStart, ColEnd - ColStart);
    list[index, 62] = list[index, 1];
    ColStart = ColEnd + 1;

    //Move one character on and repeat the process for the remaining columns
    for (int i = 1; i < 61; i++)
    {
        ColEnd = line.IndexOf(" ", ColStart);
        ColValue = line.Substring(ColStart, ColEnd - ColStart);
        list[index, i + 1] = ColValue;
        ColStart = ColEnd + 1;
    }
    list[index, 63] = "a";
}
/* Array INDEXES
* 0 = Date
* 1 = Which Forensic Mote Received it
* 2 - 61 = bytes
* 62 = also seen by
*/

for (int i = 0; i < count; i++)
{
    if (list[i, 63].CompareTo("del") != 0)
    {
        for (int j = 0; j < count; j++)
        {
            if (i != j)
            {
                bool samePacket = true;
                for (int k = 2; k < 62; k++)
                {
                    if (list[i, k].CompareTo(list[j, k]) != 0 || list[i,
                        62].Contains(list[j, 1]))
                    {
                        samePacket = false;
                    }
                }
                if (!samePacket)
            }
        }
    }
}

```

```

        {
            break;
        }
    }
    if (samePacket)
    {
        list[i, 62] = list[i, 62] + ", " + list[j, 1];
        list[j, 63] = "del";
    }
}
}
}

CreateNewRow(list, count);

file.Close();
tdt.Merge(tdt);

for (int i = 0; i < dataGridView1.RowCount - 1; i++)
{
    String CNumColour = dataGridView1.Rows[i].Cells["Seen By"].Value.ToString();
    if (CNumColour.Contains(", "))
    {
        dataGridView1.Rows[i].Cells[1].Style.BackColor = Color.Aqua;
    }
}

if (flooding == 0)
{
    flooding = count;
}

int oldFlood = flooding;

flooding += count;
flooding = flooding / 2;
if ((flooding - oldFlood) > 5)
{
    MessageBox.Show("-----Possible Flooding Occured-----\nThere were " +
        oldFlood + " packets on average for the previous minute.\nWe now have "
        + flooding + " packets per minute on average.", "FLOODING ALERT");
    if ((dataGridView1.RowCount - flooding) > 0)
    {
        for (int i = dataGridView1.RowCount - flooding; i <
            dataGridView1.RowCount - 1; i++)
        {
            dataGridView1.Rows[i].Cells[2].Style.BackColor = Color.Salmon;
        }
    }
    else
    {
        for (int i = 0; i < dataGridView1.RowCount - 1; i++)
        {
            dataGridView1.Rows[i].Cells[2].Style.BackColor = Color.Salmon;
        }
    }
}
else
{
    for (int i = 0; i < dataGridView1.RowCount - 1; i++)
    {
        String CNumColour = dataGridView1.Rows[i].Cells["Seen By"].Value.ToString();
        if (CNumColour.Contains(", "))
        {
            dataGridView1.Rows[i].Cells[1].Style.BackColor = Color.Aqua;
        }
    }
}
}
}
}
}
}

```

A.5 oWSN Field Mote

```

using System;
using Microsoft.SPOT;
using Microsoft.SPOT.Cryptography;
using Crossbow.lib.utils;
using Crossbow.platform.imote2;
using Crossbow.radio.cc2420;
using Crossbow.sensor.its400;

namespace MiddleMote
{
    public class Program
    {
        static Leds m_Leds = new Leds();

        static int BLACK = 0;
        static int RED = 1;
        static int BLUE = 2;
        static int PURPLE = 3;
        static int GREEN = 4;
        static int YELLOW = 5;
        static int TEAL = 6;
        static int WHITE = 7;

        static ushort _rfChannel = (ushort)RadioChannel.Ch11; // channel 11
        static ushort _rfPower = (ushort)RadioPower.M10DBM; // -10dbm
        static ushort _panAddr = 0xFFFF; // Pan Address
        static ushort _moteAddr = 0xFFFF; // Local Address
        static ushort _srcAddr = 0xFFFF; // Local Address
        static int _nodeid = UniqueID.SerialID;
        static int _default_nodeid = 0x01;
        static int timeout = 5000;
        static byte[] lastSeenPacket = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
            15, 15 };

        static int count = _nodeid;

        static byte[] key = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 };
        //not null, length 0 - n, 16 bytes used
        static byte[] oldKey = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 };
        //not null, length 0 - n, 16 bytes used
        static byte[] iv = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 }; // only first 8 bytes used, null or
        // less than 8 bytes are padded with zeros at end

        public static void Main()
        {
            SerialDump.print("App [MiddleMote]: Started."); // Print the applicaiton identifier

            Radio radio = new Radio(_rfChannel, _rfPower, _panAddr, _moteAddr);

            Its400 sensor = new Its400();

            if (_nodeid == 0xFFFF)
            {
                _nodeid = _default_nodeid; // set to default if there is not UID in the flash
            }

            m_Leds.set(RED);

            int round = 1;
            for (;;)
            {
                int randomNumber = Microsoft.SPOT.Math.Random(3);
                SerialDump.print("Random Number Chosen " + randomNumber + " on Execution Round " +
                    round);
                for (int i = 0; i <= randomNumber; i++)
                {
                    Observe(radio);
                }

                TransmitStats(radio, sensor);
                SerialDump.print("-----");
                round++;
            }
        }
    }
}

```



```

private static void TransmitStats(Radio radio, Its400 sensor)
{
    SerialDump.print("-----");
    SerialDump.print("-----");
    SerialDump.print("Transmitting own stats");
    m_Leds.set(PURPLE);
    if (requestToSend(radio))
    {
        string stringToConvert = _nodeid + "@ - Light - " + sensor.Light + " - Humidity - "
            + sensor.Humidity + " - Temperature - " + sensor.TempSensirion;
        byte[] packet = StrToByteArray(stringToConvert);

        Key_TinyEncryptionAlgorithm xteaData = new Key_TinyEncryptionAlgorithm(key);

        byte[] cipherBytes = xteaData.Encrypt(packet, 0, packet.Length, iv);
        //Encryption

        lastSeenPacket = cipherBytes;
        SerialDump.print("^^^^^^^^^^^^^^^^Sending DATA PACKET^^^^^^^^^^^^^^^^");
        SerialDump.print("Unencrypted DATA String: " + ByteArrayToStr(packet));
        SerialDump.print("Unencrypted DATA Packet: " + BufferToString(packet));
        SerialDump.print("Encrypted DATA Packet: " + BufferToString(cipherBytes));

        radio.Send((ushort)0xFFFF, (ushort)0xFFFF, cipherBytes);
        SerialDump.print("^^^^^^^^^^^^^^^^Sending DATA PACKET^^^^^^^^^^^^^^^^");
    }
    else
    {
        SerialDump.print("Request to send failed no data sent");
    }
}

private static bool requestToSend(Radio radio)
{
    m_Leds.set(YELLOW);
    SerialDump.print("%%%%%%%%%%%%%%%%%%%%%%%%Sending RTS%%%%%%%%%%%%%%%%%%%%%%%%");
    Microsoft.SPOT.Math.Randomize();
    string newKey = "" + Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10)
        + Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10);

    string stringToConvert = _nodeid + "@RTS#" + newKey;
    byte[] packet = StrToByteArray(stringToConvert);

    Key_TinyEncryptionAlgorithm xteaRTS = new Key_TinyEncryptionAlgorithm(key);

    byte[] cipherBytes = xteaRTS.Encrypt(packet, 0, packet.Length, iv);    //Encryption

    SerialDump.print("Unencrypted RTS String: " + ByteArrayToStr(packet));
    SerialDump.print("Unencrypted RTS Packet: " + BufferToString(packet));
    SerialDump.print("Encrypted RTS Packet: " + BufferToString(cipherBytes));

    lastSeenPacket = cipherBytes;
    radio.Send((ushort)0xFFFF, (ushort)0xFFFF, cipherBytes);
    SerialDump.print("%%%%%%%%%%%%%%%%%%%%%%%%RTS SENT%%%%%%%%%%%%%%%%%%%%%%%%");
    oldKey = key;
    key = StrToByteArray(newKey);

    for (int i = 0; i < 3; i++)
    {
        byte[] packetReceived = null;

        try
        {
            SerialDump.print("$$$$$$$$$$$$$Waiting for CT$$$$$$$$$$$$$");
            packetReceived = radio.ReceiveAny(ref _panAddr, ref _srcAddr, timeout);

            if (packetReceived != null)
            {
                m_Leds.set(YELLOW);
                SerialDump.print("Packed Received: YELLOW");
                if (AnalyzeCTS(packetReceived))

```

```

        {
            m_Leds.set(TEAL);
            SerialDump.print("$$$$$$$$$$$$CTS Received$$$$$$$$$$$$");
            return true;
        }
    }
}
catch (System.Exception ex)
{
    SerialDump.print("App [CountRecieve]: " + ex.Message);
}
}
key = oldKey;
return false;
}

private static bool AnalyzeCTS(byte[] packet)
{
    try
    {
        Key_TinyEncryptionAlgorithm xteaCTS = new Key_TinyEncryptionAlgorithm(key);
        byte[] restoredBytes = xteaCTS.Decrypt(packet, 0, packet.Length, iv); //Decryption

        string receivedPacket = ByteArrayToStr(restoredBytes);

        string packetID = receivedPacket.Substring(0, receivedPacket.IndexOf('@'));
        if ("base".CompareTo(packetID) == 0)
        {
            SerialDump.print("Unencrypted CTS String: " + ByteArrayToStr(restoredBytes));
            SerialDump.print("Unencrypted CTS Packet: " + BufferToString(restoredBytes));
            SerialDump.print("Encrypted CTS Packet: " + BufferToString(packet));

            key = StrToByteArray(receivedPacket.Substring(receivedPacket.IndexOf('#')+1));
            return true;
        }
        else
        {
            return false;
        }
    }
    catch
    {
        SerialDump.print("Malformed Packet");
        return false;
    }
}

public static void Observe(Radio radio)
{
    m_Leds.set(GREEN);
    SerialDump.print("Observeing for data: GREEN");
    byte[] packetReceived = null;

    try
    {
        SerialDump.print("Waiting for packet");
        packetReceived = radio.ReceiveAny(ref _panAddr, ref _srcAddr, timeout);

        if (packetReceived != null)
        {
            m_Leds.set(BLACK);
            SerialDump.print("Packed Received: BLACK");
            if (AnalyzePacket(packetReceived))
            {
                retransmitPacket(packetReceived, radio);
            }
        }
        else
        {
            SerialDump.print("Nothing Received: BLACK");
            m_Leds.set(BLACK);
        }
    }
    catch (System.Exception ex)
    {
        SerialDump.print("App [CountRecieve]: " + ex.Message);
    }
}

```

```

}

private static void retransmitPacket(byte[] packet, Radio radio)
{
    SerialDump.print("Retransmitting received Packet");
    m_Leds.set(WHITE);

    radio.Send(_panAddr, _moteAddr, packet);
    SerialDump.print("Sleeping 50ms after packet retransmit");
    System.Threading.Thread.Sleep(50);
}

private static bool AnalyzePacket(byte [] packet)
{
    try
    {
        SerialDump.print("Analyzing Packet: " + BufferToString(packet));
        SerialDump.print("Analyzing Packet: ");
        if (BufferToString(packet).CompareTo(BufferToString(lastSeenPacket)) == 0)
        {
            SerialDump.print("Packet has already been seen.");
            return false;
        }
        else
        {
            SerialDump.print("New Packet - Retransmit");
            lastSeenPacket = packet;
            return true;
        }
    }
    catch
    {
        SerialDump.print("Malformed Packet");
        return false;
    }
}

public static byte[] StrToByteArray(string str)
{
    return System.Text.UTF8Encoding.UTF8.GetBytes(str);
}

public static string ByteArrayToStr(byte [] dBytes)
{
    char[] temp = System.Text.UTF8Encoding.UTF8.GetChars(dBytes);
    return new string(temp);
}

private static string ByteToHex(byte b)
{
    const string hex = "0123456789ABCDEF";
    int lowNibble = b & 0x0F;
    int highNibble = (b & 0xF0) >> 4;
    string s = new string(new char[] { hex[highNibble], hex[lowNibble] });
    return s;
}

private static string BufferToString(byte[] buffer)
{
    if (buffer == null)
        throw new ArgumentNullException("buffer");
    string s = string.Empty;
    for (int i = 0; i < buffer.Length; i++)
    {
        s += ByteToHex(buffer[i]) + " ";
        if (i > 0 && i % 16 == 0)
            s += "\n";
    }
    return s;
}
}
}

```

A.6 oWSN Base Station

```

using System;
using Microsoft.SPOT;
using Microsoft.SPOT.Cryptography;
using Crossbow.lib.utils;
using Crossbow.platform.imote2;
using Crossbow.radio.cc2420;
using Crossbow.sensor.its400;

namespace BaseStation
{
    public class Program
    {
        static Leds m_Leds = new Leds();

        static int BLACK = 0;
        static int RED = 1;
        static int BLUE = 2;
        static int PURPLE = 3;
        static int GREEN = 4;
        static int YELLOW = 5;
        static int TEAL = 6;
        static int WHITE = 7;

        static ushort _rfChannel = (ushort)RadioChannel.Ch11; // channel 11
        static ushort _rfPower = (ushort)RadioPower.M10DBM; // -10dbm
        static ushort _panAddr = 0xFFFF; // Pan Address
        static ushort _moteAddr = 0xFFFF; // Local Address
        static ushort _srcAddr = 0xFFFF; // Local Address
        static int _nodeid = UniqueID.SerialID;
        static int _default_nodeid = 0x01;
        static int timeout = 5000;
        static byte[] lastSeenPacket = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
            0, 0 };

        static int count = _nodeid;

        static byte[] key = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 };
        //not null, length 0 - n, 16 bytes used
        static byte[] oldKey = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 };
        //not null, length 0 - n, 16 bytes used
        static byte[] iv = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 }; // only first 8 bytes used, null or
        // less than 8 bytes are padded with zeros at end

        public static void Main()
        {
            SerialDump.print("App [BaseStation]: Started."); // Print the applicaiton identifier

            Radio radio = new Radio(_rfChannel, _rfPower, _panAddr, _moteAddr);

            if (_nodeid == 0xFFFF)
            {
                _nodeid = _default_nodeid; // set to default if there is not UID in the flash
            }

            m_Leds.set(RED);

            for ( ; ; )
            {
                Observe(radio);
            }

            public static void Observe(Radio radio)
            {
                m_Leds.set(RED);
                SerialDump.print("Observeing for data: RED");
                byte[] packetReceived = null;

                try
                {
                    SerialDump.print("Waiting for packet");
                    packetReceived = radio.ReceiveAny(ref _panAddr, ref _srcAddr, timeout);

                    if (packetReceived != null)

```

```

    {
        m_Leds.set(BLUE);
        SerialDump.print("Packed Received: BLUE");
        if (BufferToString(lastSeenPacket).CompareTo(BufferToString(packetReceived)) !=
            0)
        {
            if (AnalyzeRTS(packetReceived))
            {
                transmitCTS(packetReceived, radio);
            }
        }
        else
        {
            SerialDump.print("Repeat of CTS packet received");
        }
    }
    else
    {
        SerialDump.print("Nothing Received: BLACK");
        m_Leds.set(BLACK);
    }
}
catch (System.Exception ex)
{
    SerialDump.print("App [BaseStation]: " + ex.Message);
}
}

private static void transmitCTS(byte[] packet, Radio radio)
{
    Key_TinyEncryptionAlgorithm xteaRTS = new Key_TinyEncryptionAlgorithm(key);
    byte[] restoredBytes = xteaRTS.Decrypt(packet, 0, packet.Length, iv); //Decryption
    string receivedPacket = ByteArrayToStr(restoredBytes);

    string packetID = receivedPacket.Substring(0, receivedPacket.IndexOf('@'));
    SerialDump.print("$$$$$$$$$$$$Sending CTS$$$$$$$$$$$$");
    Microsoft.SPOT.Math.Randomize();
    string newKey = "" + Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10) +
        Microsoft.SPOT.Math.Random(10) + Microsoft.SPOT.Math.Random(10);

    string stringToConvert = "base" + "@CTS" + packetID + "#" + newKey;

    byte[] packetCTS = StrToByteArray(stringToConvert);

    string CTSKey = receivedPacket.Substring(receivedPacket.IndexOf('#') + 1);

    oldKey = key;
    key = StrToByteArray(CTSKey);

    Key_TinyEncryptionAlgorithm xteaCTS = new Key_TinyEncryptionAlgorithm(key);
    byte[] cipherBytes = xteaCTS.Encrypt(packetCTS, 0, packetCTS.Length, iv); //Encryption

    key = StrToByteArray(newKey);

    SerialDump.print("Unencrypted CTS String: " + ByteArrayToStr(packetCTS));
    SerialDump.print("Unencrypted CTS Packet: " + BufferToString(packetCTS));
    SerialDump.print("Encrypted CTS Packet: " + BufferToString(cipherBytes));

    lastSeenPacket = packetCTS;

    radio.Send((ushort)0xFFFF, (ushort)0xFFFF, cipherBytes);
    SerialDump.print("$$$$$$$$$$$$CTS SENT$$$$$$$$$$$$");
}

private static bool AnalyzeRTS(byte[] packet)
{
    try
    {

```

```

Key_TinyEncryptionAlgorithm xteaRTS = new Key_TinyEncryptionAlgorithm(key);
byte[] restoredBytes = xteaRTS.Decrypt(packet, 0, packet.Length, iv); //Decryption
string receivedPacket = ByteArrayToStr(restoredBytes);

if (receivedPacket.IndexOf("Light") != -1)
{
    SerialDump.print("^^^^^^^^^^^^^^^^DATA RECEIVED^^^^^^^^^^^^^^^^");
    SerialDump.print("Unencrypted DATA String: " + ByteArrayToStr(restoredBytes));
    SerialDump.print("Unencrypted DATA Packet: " + BufferToString(restoredBytes));
    SerialDump.print("Encrypted DATA Packet: " + BufferToString(packet));
    return false;
}
else
{
    SerialDump.print("%%%%%%%%%%%%%%%%%RTS RECEIVED%%%%%%%%%%%%%%%%%");
    SerialDump.print("Unencrypted RTS String: " + ByteArrayToStr(restoredBytes));
    SerialDump.print("Unencrypted RTS Packet: " + BufferToString(restoredBytes));
    SerialDump.print("Encrypted RTS Packet: " + BufferToString(packet));
    string packetID = receivedPacket.Substring(receivedPacket.IndexOf('@') + 1,
        receivedPacket.IndexOf('#') - receivedPacket.IndexOf('@') - 1);

    if ("RTS".CompareTo(packetID) == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
}
catch
{
    SerialDump.print("Malformed Packet");
    return false;
}
}

public static byte[] StrToByteArray(string str)
{
    return System.Text.UTF8Encoding.UTF8.GetBytes(str);
}

public static string ByteArrayToStr(byte[] dBytes)
{
    char[] temp = System.Text.UTF8Encoding.UTF8.GetChars(dBytes);
    return new string(temp);
}

private static string ByteToHex(byte b)
{
    const string hex = "0123456789ABCDEF";
    int lowNibble = b & 0x0F;
    int highNibble = (b & 0xF0) >> 4;
    string s = new string(new char[] { hex[highNibble], hex[lowNibble] });
    return s;
}

private static string BufferToString(byte[] buffer)
{
    if (buffer == null)
        throw new ArgumentNullException("buffer");
    string s = string.Empty;
    for (int i = 0; i < buffer.Length; i++)
    {
        s += ByteToHex(buffer[i]) + " ";
        if (i > 0 && i % 16 == 0)
            s += "\n";
    }
    return s;
}
}
}

```

Appendix B: Published Papers

The papers are added in an 'as is' format in order to preserve the formatting in which the papers were published, this includes retaining original page numbering format. These papers are presented in a chronological order of date published. The first paper, which is concerned with a security protocol within wireless sensor networks, is titled “A Secure Communication Protocol for Wireless Sensor Networks.” The second paper entitled “Social engineering attack detection model: SEADM” is not directly related to wireless sensor networks but was also published during this research. The third paper entitled “Requirements for Wireless Sensor Networks in Order to Achieve Digital Forensic Readiness” relates directly to this research as it shows the list of requirements in order to achieve digital forensic readiness in a wireless sensor network. The fourth paper, entitled “A prototype for achieving digital forensic readiness on wireless sensor networks” is also directly related to this research as it was used to show that having a set of requirements is extremely helpful when trying to implement digital forensic readiness in wireless sensor networks.

These papers start on the subsequent page.



Security, Assurance and
Privacy: Organizational
Challenges
Proceedings of the
8th Annual Security
Conference
*Discourses in Security
Assurance & Privacy*
Las Vegas, NV, USA
April 15-16, 2009

Editor
Gurpreet Dhillon
Virginia Commonwealth
University, USA

ISBN: 978-1-935160-04-5

www.security-conference.org

A Secure Communication Protocol for Wireless Sensor Networks

Francois Mouton, HS Venter
University of Pretoria

Abstract

The field of wireless sensor networking is still a new and upcoming one and unfortunately still lacking in terms of security. All communications between different nodes (motes) are sent out in a broadcast fashion. These broadcasts could easily be intercepted by any adverse mote in the vicinity. This paper proposes a security protocol which can be implemented in current wireless sensor networks. The proposed protocol relies on symmetric encryption where the encryption key changes constantly throughout the communication. Furthermore we evaluate the efficiency of the proposed security protocol against some known attacks on wireless sensor networks in order to show how such attacks can be thwarted.

Keywords: communication, encryption, security, wireless sensor networks, XTEA.

I Introduction

Striving towards a better lifestyle has lead to a great improvement in the technology we have access to in today's world. These technologies should be implemented in such a way that almost anyone can use them with little or no effort. This is partially due to the fact that many of the technologies we use today have converged to a single device. As an example, mobile phones have advanced from simply being communication items to almost being fully functional computers. The rapid adoption of mobile phones has lead to the development of next-generation networks (NGNs) (Orecchia et al 2004; Santi 2005; Tseng et al 2003) which provide a new platform for many different applications of wireless and ad-hoc networks in a ubiquitous environment. Wireless sensor networks (WSNs) form part of NGNs and form the main focal point in this paper.

In general, all of the above-mentioned technologies were developed to help and improve our ability to accomplish our tasks on a daily basis. They have, unfortunately, also become some of the main targets for people who have ideas of malicious and mal intent on their agendas. In addition, when it comes to wireless, there is a greater risk as data is transferred over the air between devices, which make them more susceptible to data interception than tapping into a physical network. In addition, the data which might be intercepted raises a confidentiality issue, discouraging certain people from using these wireless devices.

Wireless sensor networks inherit many of the aforementioned problems. The nodes in a wireless sensor network can be referred to as either sensors or motes. In this paper they are referred to as motes. Wireless sensor networks are vulnerable to the attacks of those who want to steal valuable data generated by the motes. The motes typically broadcast the data to other motes in the vicinity. The receiving motes continue broadcasting the data until it is able to reach the gateway, also referred to as the base station. The data is then sent to a central server (Orecchia et al 2004; Slijepcevic et al 2002). It is difficult to determine whether there is another wireless sensor network in the area that is eavesdropping on the data being transferred.

Another huge problem with wireless sensor networks is their physical design. It consists of small objects (motes) which are scattered or strategically placed throughout a specific

area. These motes could be physically removed or damaged by someone moving around in the area. The motes are also dependant on a battery which powers the device. Even if the motes are physically secured, attackers could attempt to force communication with the motes on a continuous basis. This would drain the battery power and cause the motes to become unresponsive.

This paper does not address all the above-mentioned problems, however, the purpose of this paper is to identify known attacks on wireless sensor networks and suggest a security protocol which would make a wireless sensor network less vulnerable and susceptible to these types of attacks.

The remainder of the paper is structured as follows. The second section provides background in terms of wireless sensor networks and known attacks on wireless sensor networks. Section three discusses the proposed security protocol and how it can be implemented. The fourth section contains a discussion about the effectiveness of the security protocol against possible attacks we have proposed. Finally, the last section concludes with a summary of the security protocol and future work.

2 Background

Wireless Sensor Networks (WSNs) is still a relatively new area of research in Computer Science. The first papers on WSNs only appeared around the start of the 21st century. Most of the research on WSNs has been on new areas where WSNs can be applied to modern lifestyles. Before suggesting a solution to the security issue, this paper will provide background information necessary for a better understanding of WSNs.

2.1 Wireless Sensor Networks

WSNs belong to the general family of sensor networks that use multiple distributed sensors to retrieve data from various environments of interest. Chong and Kumar (2003) provide a history on previous accomplishments of WSNs. He also shows how WSNs have evolved in terms of sensing, communication and computing. WSNs consist of wireless nodes with embedded processors, ad-hoc networks (Estring et al 2001) and wireless communication (Ye et al 2002). The authors summarise the concept of WSNs to be an ad-hoc network which consists of tiny and resilient computing nodes known as motes or sensors. These motes are extremely efficient with regard to power consumption and can collaborate with other motes within their vicinity effectively. A graphical representation of a wireless sensor network is provided in fig.1 and the functions of each of the components are briefly summarised in Table 1 (Heinzelman et al 1999; Sohrabi et al 2002).

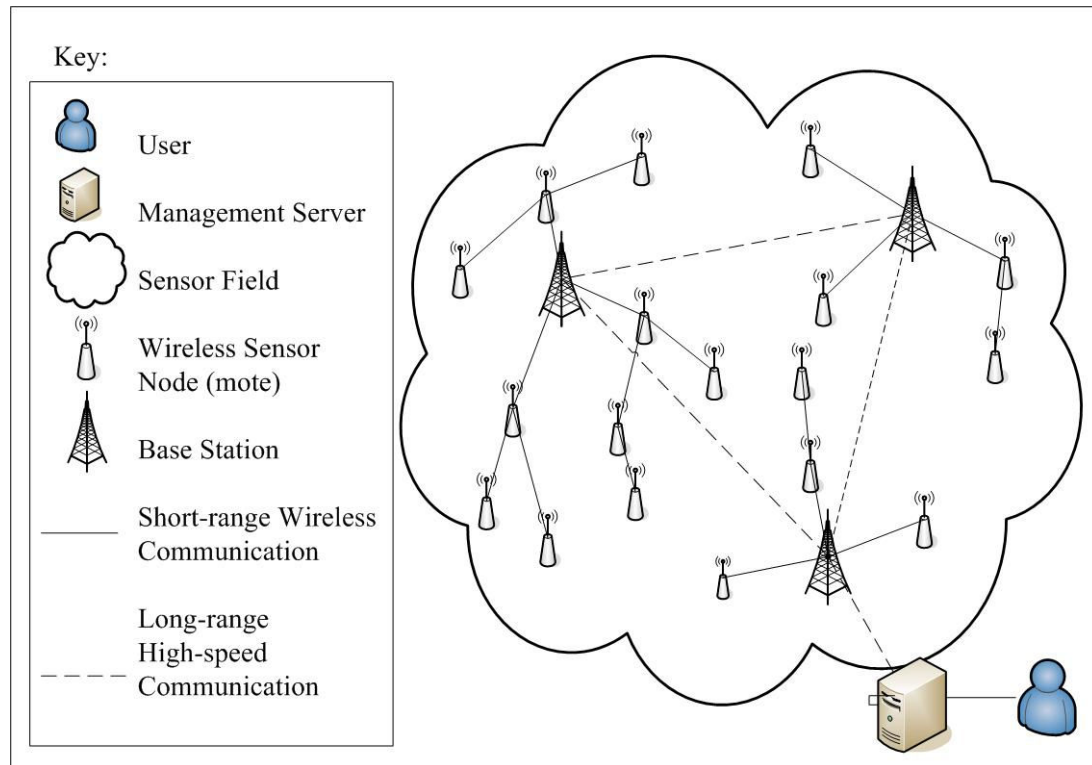


Fig. 1. A graphical representation of a wireless sensor network.

Table 1. Summary of the functions of each component in Fig.1.

WSN component	Brief summary of the component
User	The user can interact with the WSN through the management server.
Management Server	The management server serves as an interface console for the WSN.
Sensor Field	The sensor field denotes the physical boundaries of the WSN.
Wireless Sensor Node (mote)	Each mote contains a small subset of the various sensors. Motes in the network can also act as repeaters for packets which need to reach the base station.
Base Station	A base station serves as a gateway-node through which the information of the motes has to travel to reach the management server.
Short-range Wireless Communication	Short-range wireless communication links are established between neighbouring motes and the neighbouring base stations.
Long-range High-speed Communication	Long-range high-speed communication links are established between further-ranged base stations and the management server.

WSNs can be used in many environments. Motes may consist of many different types of sensors, such as thermal, visual, infrared, radar or acoustic. These motes can then monitor a wide variety of ambient conditions, including humidity, pressure, sound, noise levels, temperature, lightning conditions and objects moving through a designated area (Elson and Estrin 2001; Kahn et al 1999).

Some applications of WSNs include military applications such as tracking moving objects and battlefield surveillance (Zhao et al 2002). Environmental applications include habitat monitoring, forest fire detection and flood detection (Mainwaring et al. 2002). Health applications include tracking and monitoring doctors and patients in hospitals and drug administration in hospitals (Lu et al 2002). Finally, WSNs also include home and building automation applications.

The next subsection focuses on introducing the reader to known attacks on wireless sensor networks.

2.2 Known attacks on wireless sensor networks

There are quite a number of known attacks which can render the entire network ineffective (Perrig et al 2004). The focus will be on four of these attacks, namely data interception, flooding attacks, sinkhole attacks and attacks which require physical access to the motes. These attacks are briefly discussed in the next subsections.

2.2.1 Data Interception

Each mote broadcasts the information it has gathered from its ambient surroundings in an attempt to reach the other motes. Currently, as far as the authors are aware, there is no encryption on the data which is broadcasted throughout the WSN. This allows a hostile entity to plant its own WSN in the surrounding area. The hostile WSN can then eavesdrop on any communication and will pass on any intercepted information to the hostile management server (Slijepcevic et al 2002). This could disclose valuable information of the owner of the legitimate WSN to the hostile entity.

2.2.2 Flooding Attacks

WSN motes are by default set to accept any incoming traffic. It takes this information and then processes it in order to determine what should be done with it. The receiving and processing of the data drain valuable battery power from the mote.

If a hostile entity injects a hostile WSN into the range of the legitimate WSN, a flooding attack can then be launched on the legitimate WSN. A flooding attack is an attack which constantly sends out information to all the motes in its vicinity. This type of attack could also be seen as a DoS (denial of service) attack (Perrig et al 2004), because the attack would drain the motes in the legitimate WSN of their battery power. This would render the motes useless and they would have to be replaced with a new set of motes with fresh batteries.

2.2.3 Sinkhole Attacks

This is one of the most difficult attacks to deal with when using wireless sensor networks (Zhu et al 2006). In a sinkhole attack, an adverse mote will try to attract

information from its neighbours (Karlof and Wagner 2003). It would then either pass this information on to a hostile entity or just drop the packets. This would lead to a DoS attack on the legitimate WSN.

This attack is accomplished by a hostile mote broadcasting a message to the neighbouring motes, that itself is the closest to the base station. All the motes in that vicinity would then assume that this information is correct and would not attempt to send information to other motes or the base station, but rather to the hostile mote. This would allow the hostile mote to then receive all the network packets in the vicinity of other motes, causing legitimate information never to reach the base station.

2.2.4 Attacks Which Require Physical Access to the Motes

Most research articles that we have consulted have very little emphasis on these types of attacks. It is very important to note that an intruder could physically connect his workstation to a mote, once found, and transfer all the code which is on the mote directly to his workstation using an appropriate interface.

An attacker could even then take the code he received, modify it and deploy it back onto the mote. This is a very powerful attack as it appears as if the legitimate mote is sending legitimate communication to other legitimate WSN motes, while it is in fact falsified communication injected by an attacker.

In the next section the proposed security protocol will be introduced and explained, after which a discussion follows on how the above-mentioned attacks can be thwarted.

3 Proposed WSN Security Protocol

The proposed security protocol consists of an implementation of the XTEA (eXtended Tiny Encryption Algorithm) and a new way of having dynamic encryption keys in the network. A prototype of the protocol was developed and demonstrated on an existing WSN consisting of Crossbow Imote2 sensor motes. The remainder of this section is dedicated to introducing the Imote2 sensor motes, the XTEA algorithm and an explanation of how the proposed security protocol works.

3.1 Imote2 Sensor Motes

The Crossbow Imote2 is an advanced sensor network node platform designed for demanding wireless sensor network applications that require high CPU performance and reliability. Each sensor node in the prototype consists of a battery board, an Imote2 board and an ITS400 sensor board. The base station only consists of an Imote2 board as it is powered by a USB port.

3.2 eXtended Tiny Encryption Algorithm

The eXtended Tiny Encryption Algorithm (XTEA) is an extension of Tiny Encryption Algorithm (TEA) which is a very simple cryptographic algorithm. TEA is a very small algorithm and it uses very little resources and processing power. The extended version of TEA is a very good cipher when a relatively high level of security is required and when computational power is limited (Blom et al). The cipher was designed by Roger Needham and David Wheeler of the Cambridge Computer Laboratory, and the algorithm was presented in a technical report in 1997 (Hong et al 2004).

It is important to note that XTEA is a symmetrical encryption algorithm. This means that the key that was used to encrypt a packet must also be used to decrypt the same packet. Thus, both parties involved in the communication must have the encryption key. This poses a problem in WSNs because motes are vulnerable when they can be physically picked up by an attacker. This allows an attacker to monitor the traffic throughout the network and extract the key from the mote to decrypt the traffic which has been gathered. The protocol proposed in this paper, however, specifically addresses this issue and the protocol is explained in further detail in the next section. Another classic problem with symmetric key encryption is the key distribution problem, however, the authors assume that, due to the nature of a WSN, keys are distributed on the motes when they are scattered or strategically placed. Hence, no initial key distribution has to take place.

3.3 Protocol Implementation

The protocol which is presented in this paper relies heavily on the use of request-to-send (RTS) and clear-to-send (CTS) operations. Every message passing throughout the network would be encrypted using XTEA. The key used for encryption would be changed as packets are sent back and forth to prevent an attacker to use a ciphertext-only attack. This attack occurs when an attacker is able to collect a large amount of ciphertext and deduce from that the key or the plaintext which was transmitted. The protocol must not remove the main functionality of a WSN of being an ad-hoc network where extra motes can be added dynamically to the network. In the following subsections an explanation of the protocol is provided wherein each of the main functions of the protocol are discussed separately.

3.3.1 Broadcasting packets throughout the network

The protocol does not save any predefined routes to any mote anywhere in the network. It has been implemented in such a fashion specifically to avoid sinkhole attacks. The network is immune to a rouge mote if the rouge mote attempts to act like a sinkhole to broadcast information that it is the mote nearest to the base station. Every packet that is sent in the network will be broadcasted from the mote which sent it, to all its neighbouring motes. This has the possibility to introduce extra overhead to the network, but this is addressed in how we handle repeating packets in section 3.3.6. The first packet that a mote will broadcast throughout the network is a RTS packet. This type of packet will be discussed in detail in the following section.

3.3.2 Request to send

The RTS packet consists of three fields: the mote ID which is unique to each mote in the network, a RTS flag which indicates that this packet is indeed a RTS packet, and a randomly generated key which is afterwards used to encrypt the CTS packet. The generated key is referred to as the CTSKey. It is generated by using a random number generator which is built into the hardware of the mote. We assume that the random number generator contains enough entropy to generate a random-like sequence that is not predictable in a feasible amount of time. After the RTS has been sent, the mote moves into an idle state where it is listening for a CTS packet. The amount of time to wait for the CTS can be manually defined. In an

experimental setup a time of ten seconds is the maximum threshold to wait. If it receives a CTS it will send a data packet which is discussed in section 3.3.4, otherwise it will go into an observe mode which is discussed in section 3.3.6.

The packet that is sent can be formulated to resemble fig.2 where the data inside the brackets is encrypted with the initial mote key (MoteKey v1.1), which is before the brackets:

moteID + MoteKey v1.1 (RTS + CTSKey)

Fig. 2. The RTS Packet

The next section explains how the CTS packet is constructed and transmitted.

3.3.3 Clear to send

The base station is the only mote that is able to interpret CTS packets. This is due to the fact that only the base should be allowed to capture any data generated on the network and be able to decipher it.

The network is also immune to rogue motes attempting to send fake CTS packets, as a true CTS packet will be encrypted. Thus, for a rogue mote to send a fake CTS packet, it would need to know what encryption scheme and key to use.

The base station decrypts the RTS packet using MoteKey v1.1 for the particular mote ID. Note that each base station is initially preloaded with a list of the initial mote keys. The base station interprets the RTS packet and determines from which mote it originated in the network. It then looks up if the mote ID corresponds to a mote ID that is supposed to be in the network and after this verification has succeeded it then sends out the CTS packet. The CTS packet can, therefore, only be sent by the base station if a valid RTS packet was received by the base station.

The CTS packet consists of the base station ID, a CTS flag to indicate it is indeed a CTS packet and a randomly generated key which will be used to transmit data towards the base station in future communication with the base station. This packet is encrypted using the randomly-generated CTS key which was received in the RTS packet.

The packet that is sent can be formulated to resemble fig.3 where the data inside the brackets is encrypted with the key which is before the brackets:

CTSKey(BaseID + CTS + MoteKey v1.2)

Fig. 3. The CTS Packet

The next section explains how each data packet is constructed and transmitted.

3.3.4 The data packet

The mote that sent off the RTS packet is still in a state of waiting for the CTS. When it receives the CTS it uses the CTSKey it generated to decrypt this packet. If the CTS packet was successfully decrypted then it is indeed a valid CTS packet. The mote then determines if the base station ID, as received in the CTS packet, corresponds with the base station ID the mote has previously been authenticated with.

If all three of these conditions are met, the mote reads MoteKey v1.2 from the CTS packet and then replaces its mote key with this new mote key. This process is further explained in section 3.3.5. The data packet is then constructed with information gathered from the ITS400 sensor board.

In our implementation we retrieved the light, humidity and temperature data from the sensor board. The data is then constructed into such a data packet. At the front of the data packet the mote ID is added again. The packet is then encrypted with the new mote key and sent back to the base station.

The data packet that is sent can be formulated to resemble fig.4 where the data inside the brackets is encrypted with the key precedes before the brackets:

MoteID + MoteKey v1.2(Data)

Fig. 4. The Data Packet

Enough background is given for us to summarise and discuss the entire encryption protocol and how the keys are changed throughout all the packet transfers described in the previous sections. We repeat some of the previously mentioned concepts for the sake of clarity.

3.3.5 The encryption key handling

The encryption key handling is explained in fig.5.

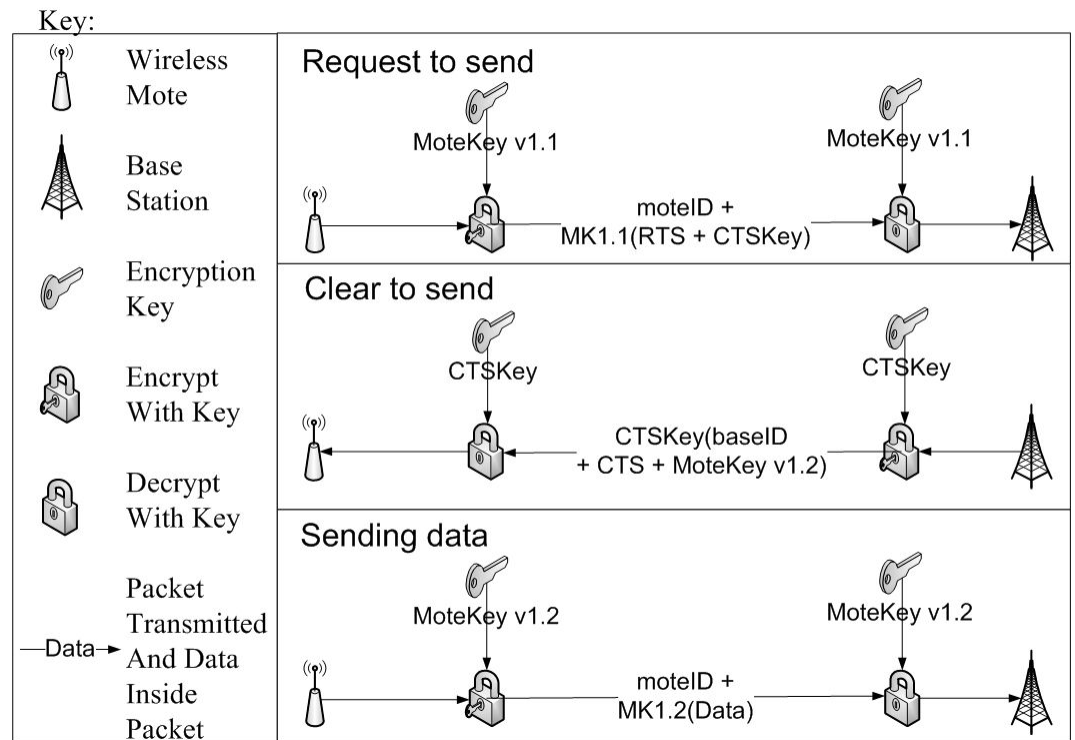


Fig. 5. Encryption key handling

Before the WSN is deployed into the sensor field, each mote is preloaded with an encryption key. This key is referred to as MoteKey vX.Y. The vX.Y represents the following: X is the mote ID and Y is the version number of the key, indicating how many times the MoteKey has been changed. For this example we can assume that we are

using mote ID I and the initial version of the key, i.e. Y is equal to I . The base station has a list of all the motes that are deployed into the sensor field with each mote's associated initial MoteKey (i.e. MoteKey v.X.I for each mote). Each mote can, therefore, be uniquely identified on the base station by using the mote's ID to correspond which MoteKey belongs to each mote.

In the RTS phase a random CTSKey is generated. The RTS packet then contains all the RTS related information and the CTSKey. This packet is then encrypted using MoteKey v1.1 before it is broadcasted into the WSN. The mote ID is not encrypted though, but is sent along with the RTS packet. The mote which sent the packet also temporarily saves the CTSKey.

Upon receipt of the RTS packet by the base station, it analyses the mote ID and uses the corresponding MoteKey to decrypt the packet. In this case it would be MoteKey v1.1. If the RTS packet was decrypted successfully, the base station generates a new MoteKey. This key would now be known as MoteKey v1.2. The base station also extracts the CTSKey from the RTS packet. After this key has been extracted, the CTS packet is constructed and the MoteKey v1.2 is added onto the packet. The packet is then encrypted using the CTSKey and is then sent back to the mote that sent the RTS. MoteKey v1.2 is only stored in a temporary slot and MoteKey v1.1 is not yet replaced.

When the CTS packet arrives at the mote, it tries to decrypt the packet with the CTS Key. In the case that it was unsuccessful, it means that the mote received a packet in the network which is actually not a CTS packet meant for the particular mote. It does however still continue to wait for the predefined threshold, i.e. 10 seconds, for the real CTS packet. In the case that it was successful, it analyses the base station ID and determines if this was the intended base station ID it sent the original RTS packet to. If the base station ID is indeed correct, the mote then tests the mote ID and the CTS signature as discussed before. The mote then checks for the new MoteKey. It then replaces MoteKey v1.1 with MoteKey v1.2. This new key is only written to the RAM on the mote. If the mote is switched off or has to reboot for whatever reason, only MoteKey v1.1 will be available. Thus, if the mote reboots, it will be disconnected from the network for security reasons and will be discussed later. The Data packet is then generated by using the data received from the ITS400 sensor board. The data packet consists of the mote ID and the sensor data which has been encrypted using MoteKey v1.2. The data packet is then transmitted to the base station.

The base station then receives this data packet and determines which mote it came from by looking at the mote ID. At this stage the base station is not able to determine if this is a RTS or a data packet. It first attempts to decrypt the packet using MoteKey v1.2 which has been stored in temporary storage (in the base station's RAM). If the base station was able to decrypt the package successfully, only then does it replace MoteKey v1.1 with MoteKey v1.2 as this will mean that the CTS packet successfully arrived at the mote, which originally sent the RTS packet. However, if the base station was unable to decrypt the packet, it then resorts back to MoteKey v1.1 to attempt to decrypt it. If it is successful with MoteKey v1.1, it is an indication that the CTS did not arrive in tact at the mote. The packet which has then been received would be another RTS packet. The base station would then send a new CTS packet. The sensory data that is decrypted by the base station is then directly sent to the management server. This process continues every time new sensory data is sent from a mote to the base station.

Using the protocol described above, we are able to create a secure channel between each mote in the field and the base station. Every packet that is sent throughout the network will effectively be encrypted with a different key.

3.3.6 Repeating packets

Every mote in the network spends most of its time only observing packets that go around in the network. The motes only actually transmit their sensory data on specified time intervals.

A mote is in an observation phase when it is capturing packets that are generated by other motes in the WSN. A mote first tests if the packet that has been received has already been seen by comparing the packet to previously received packets. If the packet has already been seen, the mote will not retransmit it again. If the mote retransmits the packet that has been previously seen again, the possibility exists that the entire network might get flooded and no communication could occur further on.

In the scenario where the packet has not been seen before, the mote retransmits the packet to all of its neighbouring motes. The neighbouring motes will then test if they have seen the packet and if not, they will retransmit the packet further. If two packets are received at the exact same time, they are put into a queue and analysed sequentially by the receiving mote.

3.3.7 Handling network failures

The protocol shows promising resilience to errors. If a packet cannot be interpreted by a mote, it is most likely a malformed packet. The mote simply drops the packet and switches to observe mode. The main concern we have with the protocol is when the encryption keys go out of sync (i.e. a mote is disconnected from the network e.g. by rebooting it, running out of power etc.). The encryption key on the mote will only be updated if the base station has successfully responded with a CTS. We can assume that if there a path existed to the base station, there would always be a path back to the mote.

The protocol is also resilient against single motes failing in the network as long as there is still a possible path to the base station. Due to the way the packets are broadcasted we are always assured that packets will reach the destination if a path exists to the destination.

3.3.8 Redeploying motes or adding new motes to the network

As we have discussed earlier, if a mote reboots for whatever reason or the batteries need are replaced, the mote will lose all communication to the network as the keys will be out of sync. This technique, however, has been implemented as a security precaution. It allows us to remove the vulnerability where someone can physically take the mote and reprogram some extra data onto it which could be malicious to our network. The mote that has been tampered with will not be able to communicate to our base station. If a mote has to be redeployed or a new mote has to be added, it must first be assigned a MoteKey vX.I. This key must also be added to the base station's list of keys. This technique allows us to easily control which motes are allowed into the network and which are not. We can also use this to remotely disable communication of certain motes to the base station, if necessary (i.e. if a mote was compromised). If the base station does not know the MoteKey of a mote, it will reject any communication coming from the mote.

4 Discussion of the Proposed Security Protocol

This section discusses the use of the proposed security protocol in an existing WSN. A graphical representation of a scenario of how the security protocol was implemented in a prototype scenario can be seen in fig.1. In the following subsections explain how each of the attacks mentioned in the background section can be prevented using our proposed protocol. In addition, a discussion on the limitations of the protocol is also provided here.

4.1 Flooding Attack

In the opinion of the authors, flooding attacks are one of the most difficult attacks to thwart in a WSN. We have physically simulated a flooding attack on a WSN using our protocol. The results are promising, because the WSN was very resilient to the flooding attack.

We simulated the flooding of a WSN by having the motes transmitting sensor data faster than it usually would. During normal operation of the WSN, however, each mote waits for a predefined number of packets to pass through it or a specified timeframe before it decides to transmit its own sensor data. In the case where a flooding attack is present this waiting cycle is almost nullified. There is however a slight benefit to this. This will cause the part of the network which is being flooded to send data faster and we will be able to determine where the flooding attack originates from. This allows us to physically go and eliminate the rogue mote in the network which is flooding our network. According to the authors, the only successful way to fully eliminate a mote that is flooding a WSN is to physically go into the sensor field and remove it.

4.2 Sinkhole Attack

The sinkhole attack is a very devastating attack on a network and, as shown previously, a single sinkhole has the ability to fully disable a network. The protocol we are proposing is, however, very effective against a sinkhole attack. Since the protocol does not rely on a predefined route to the base station, a sinkhole which propagates a route to the base station will have no effect on the network.

This makes our protocol one of the best countermeasures to sinkhole attacks in WSNs.

4.3 Data Interception

In our protocol we use end-to-end encryption. This means that any data which is sent over the network towards another mote will be encrypted through all the motes it has to pass in order to get to the destination.

As we have discussed before, the protocol is also very resilient against a cipher text attack. This is due to the fact that we change the encryption key on every transmission. We are aware that there is a potential flaw with the way in which we apply the encryption. The problem arises when an attacker captures all the traffic in the network and then might be able to decrypt a single packet from that traffic using a brute-force attack. This might allow the attacker to be able to decrypt a whole chain of communication between a single mote and the base station if all the traffic has been captured. We have, however, not considered this to be a great concern. For an attacker to brute-force a single packet, it might take an extended time, such as months or years.

The attacker needs to capture all communication during this timeframe, which is practically infeasible.

For example, if we were using AES to crack a single key, all the traffic can be decrypted from that point onwards. However, this is true in almost any encryption scheme available up to date. Therefore, research into this is beyond the scope of this paper.

4.4 Attacks Which Require Physical Access to the Mote

The main issue we are dealing with here is that an attacker could connect his workstation to a mote and retrieve all the code which was imbedded on the mote.

This is actually a very critical issue as encrypted keys could be stored on the mote. The attacker could also only slightly modify the operation of the mote and cause it to become a hostile mote to the network. This is also a very difficult attack to avoid as motes are normally scattered in a sensor field and is not monitored that often. Also the mote that the attacker uses might have been taken from our network, modified, and redeployed into the field.

This issue is, however, addressed by the protocol since any mote that is redeployed into the network has to be provided with a new encryption key for entry into the network. Any mote that has been powered down will not be able to communicate to the base station until initial authentication takes place again.

In the next section we discuss the throughput and the known limitations of our protocol.

4.5 Throughput and Limitations of the Security Protocol

It is the opinion of the authors that the main concern with any proposed protocol is whether it will be able to handle large amounts of network traffic. Most WSN applications need to transfer data in almost real time. This is the case where WSNs are used to track objects travelling in a sensor field or where it monitors for forest fires or any other natural disaster. These applications are often time critical applications and they can also not afford to lose any single packet in the case of an emergency.

We have stress-tested the protocol and have found that we can send about 6 to 8 data packets from a single mote per second. This would estimate to 18 to 24 packets exchanged on the network per second. We are quite confident that we can classify this as almost real time communication and would be sufficient for WSN applications.

However, it is also very important that we address the limitations of the protocol. The main limitation we have with the protocol is the physical effort it takes to deploy new motes into the network or replace the batteries on motes. If a new mote is deployed or batteries have to be replaced, a new encryption key has to be generated and manually added to the base station before this mote can communicate back to the base station.

Conclusion

Wireless sensor networks currently have very little security mechanisms embedded into them and are susceptible to a myriad of attacks, especially denial-of-service attacks. This paper proposed a security protocol which can be implemented in WSNs in order to improve WSN security.

This paper mainly focused on thwarting flooding attacks, sinkhole attacks, data interception and attacks which could occur if an attacker had physical access to the motes. In the opinion of the authors, we believe that there are room for improvement on preventing flooding attacks in WSNs. Flooding attacks are one of the most difficult attacks to thwart, because extensive processing power is required to analyse packets and determine that it is a flooding attack. If we simply start rejecting packets after a certain threshold, we could be the cause of a self-inflicted denial-of-service attack on our own network, should legitimate packets be wrongfully rejected.

The security protocol that was proposed used XTEA to perform end-to-end encryption on the network. Encryption keys were also exchanged using this algorithm. It provided us with the functionality that data which were intercepted could not be decrypted easily and the network was fully resilient against sinkhole attacks. It also provided more efficient security in terms of redeploying motes or adding new motes to the WSN.

This paper showed that the proposed WSN security protocol is resilient against most of the known attacks on wireless sensor networks and would be of benefit to any existing WSN which has little or no imbedded security.

In further research the authors will explore finding additional countermeasures for other types of attacks on wireless sensor networks. In addition, the authors also plan to explore the possibilities of injecting a second WSN as a forensic WSN. Such a forensic WSN might also act as an intrusion detection system in a WSN environment.

References

- Blom S, Kleiboer L, Stegeman R, Willcox S., "Power consumption of the xTEA and RC5 cryptographic algorithms,"
- Chong, C., Kumar, S.P. (2003) "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, 91 (8): 1247-1256.
- Elson, J., Estrin, D. (2001), "Time synchronization for wireless sensor networks," *Parallel and Distributed Processing Symposium., Proceedings 15th International: 1965-1970.*
- Estrin, D., Girod, L., Pottie, G., Srivastava, M. (2001), "Instrumenting the world with wireless sensor networks," *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, 4: 2033-2036.
- Heinzelman, W. R., Kulik, J., Balakrishnan, H. (1999), "Adaptive protocols for information dissemination in wireless sensor networks," *In Proceedings of the 5th Annual ACM/IEEE international Conference on Mobile Computing and Networking (Seattle, Washington, United States, August 15 - 19, 1999). MobiCom '99. ACM, New York, NY: 174-185.*
- Hong S, Hong D, Ko Y, Chang D, Lee W, Lee S. (2004), "Differential Cryptanalysis of TEA and XTEA," *Information Security and Cryptology - ICISC 2003*, 2971/2004: 402-417.
- Kahn, J. M., Katz, R. H., Pister, K. S. (1999), "Next century challenges: mobile networking for "Smart Dust"," *In Proceedings of the 5th Annual ACM/IEEE*

- international Conference on Mobile Computing and Networking MobiCom '99*. ACM, New York, NY: 271-278.
- Karlof, C., Wagner, D. (2003), "Secure routing in wireless sensor networks: Attacks and countermeasures," *In Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*: 293-315.
- Lu C., Blum, B.M., Abdelzاهر, T.F., Stankovic, J.A., He, T. (2002), "RAP: a real-time communication architecture for large-scale wireless sensor networks," *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*: 55-66.
- Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. (2002), "Wireless sensor networks for habitat monitoring," *In Proceedings of the 1st ACM international Workshop on Wireless Sensor Networks and Applications. WSNA '02*. ACM, New York, NY: 88-97.
- Orecchia, L., Panconesi, A., Petrioli, C., and Vitaletti, A. (2004), "Localized techniques for broadcasting in wireless sensor networks," *In Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing. DIALM-POMC '04*. ACM, New York, NY: 41-51.
- Perrig, A., Stankovic, J., and Wagner, D. (2004), "Security in wireless sensor networks," *Commun. ACM* 47: 53-57.
- Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J. (2001), "SPINS: Security Protocols for Sensor Networks," *In Seventh Annual ACM International Conference on Mobile Computing and Networks (Mobicom 2001), Rome Italy*: 521-534.
- Santi, P. (2005), "Topology control in wireless ad hoc and sensor networks". *ACM Comput. Surv*, 37 (2): 164-194.
- Slijepcevic, S., Potkonjak, M., Tsiatsis, V., Zimbeck, S., Srivastava, M.B. (2002), "On communication security in wireless ad-hoc sensor networks, Enabling Technologies: Infrastructure for Collaborative Enterprises," *WET ICE 2002. Proceedings. Eleventh IEEE International Workshops on*: 139-144.
- Sohrabi, K., Gao, J., Ailawadhi, V., Pottie, G.J. (2000), "Protocols for self-organization of a wireless sensor network," *Personal Communications, IEEE*, 7 (5): 16-27.
- Tseng, Y., Ni, S., Shih, E. (2003), "Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network," *IEEE Transactions on Computers*, 52 (5): 545-557.
- Ye, W., Heidemann, J., Estrin, D. (2002), "An energy-efficient MAC protocol for wireless sensor networks," *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3: 1567-1576.
- Zhao, F., Shin, J., Reich, J. (2002) "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Mag.*, 19: 61-72.
- Zhu, S., Setia, S., and Jajodia, S.: LEAP+ (2006), "Efficient security mechanisms for large-scale distributed sensor networks," *ACM Trans. Sen. Netw.* 2 (4): 500-528.

Social Engineering Attack Detection Model: SEADM

Monique Bezuidenhout

Department of Psychology
University of Pretoria
Pretoria, South Africa
monique.bezuidenhout@up.ac.za

Francois Mouton

Department of Computer Science
University of Pretoria
Pretoria, South Africa
moutonf@gmail.com

H.S. Venter

Department of Computer Science
University of Pretoria
Pretoria, South Africa

Abstract—Social engineering is a real threat to industries in this day and age even though the severity of it is extremely downplayed. The difficulty with social engineering attacks is mostly the ability to identify them. Social engineers target call centre employees, as they are normally underpaid, under skilled workers whom have limited knowledge about the information technology infrastructure. These workers are thus easy targets for the social engineer. This paper proposes a model which can be used by these workers to detect social engineering attacks in a call centre environment. The model is a quick and effective way to determine if the requester is trying to manipulate an individual into disclosing information to which the requester does not have authorization for.

Keywords: Social engineering, social psychology, emotional state, information sensitivity.

I. INTRODUCTION

Social engineering, in this context, refers to various techniques that are utilized to obtain information in order to bypass security systems, through the exploitation of human vulnerability [1]. As clearly stated by various authors [2], [3], [4], [5], the human element is the ‘glitch’ or vulnerable element within security systems. It is the basic ‘good’ human natured characteristics that make people vulnerable to the techniques used by social engineers, as it activates various psychological vulnerabilities, which could be used to manipulate the individual to disclose the requested information [2], [5].

Individuals make themselves even more vulnerable to social engineering attacks by not expecting to ever be a victim of such an attack, and many will never know that they were a victim of such an attack. The majority of the public are not aware of this technique, and do not fully comprehend the extent to which these techniques to obtain information, can be used, and the potential it holds for dire personal, economic and social consequences and losses for the individual and institution. An individual may believe that the information they possess is of no particular value to another person, nor could it be used for any malicious act, and will thus be more willing to disclose information freely. However, the social engineer is dedicated to researching various aspects and gathering

information from various sources. Combined the acquired information can have dire consequences.

On the other end of the spectrum, the individual might believe that they will not fall prey to such an attack, as they would be able to recognize such an attack instantaneously. However, the social engineer is a skilled human manipulator, preying on human vulnerabilities using various psychological triggers that could foil human judgment.

The problem is to successfully detect social engineering attacks whilst working in a stressful environment, where decisions must be made instantaneously. It is for this reason that a practical model, that can be easily implemented and used by all levels of employees, is necessary and proposed within this paper. This model should be used in combination with training on various social engineering techniques, the psychological vulnerabilities it may elicit, and institutional policies and procedures.

The two main perspectives of social engineering - the psychological perspective and the computer science perspective - are accounted for within this model. The psychological perspective focuses on the emotional state and cognitive abilities of the individual. The computer science perspective focuses on information sensitivity, one of the cornerstones of information security. Other important factors incorporated within this model are the urgency of requested information and an individual’s comprehension of the requested information.

The remainder of the paper is structured as follows. Section 2 provides background about Social engineering, and Section 3 discusses background on the process of human reasoning and decision-making. Section 4 introduces the proposed model, as presented within this paper, which was developed for social engineering attack detection and provides an in-depth discussion of each of the pertinent elements of the model. Section 5 provides scenarios in order to demonstrate the effectiveness of the model. Finally, Section 6 concludes with a summary of the social engineering attack detection model and suggested future work.

II. SOCIAL ENGINEERING

According to [1] social engineering is defined as the techniques used to exploit human vulnerability to bypass security systems in order to gather information. As indicated by this definition, social engineering attacks imply interaction with other individuals, indicating the psychological aspect of social engineering.

Various psychological vulnerabilities and triggers, used by social engineers, have been identified, which aim to influence the individual's emotional state and cognitive abilities in order to obtain information. To successfully defend against these psychological triggers, the individual will need to have a clear understanding of these triggers in order to recognize each during a social engineering attack. Seven psychological vulnerabilities has been defined by [6]. These psychological vulnerabilities are the following [2],[3],[7],[8]:

Strong Affect: When a strong emotion is triggered, such as anger, excitement, fear or anxiety, an individual's cognitive ability may be seriously hampered. This may include their ability to make decisions rationally, evaluate the situation, make counterarguments, and reason logically, which is why this is such an effective technique used by social engineers [8]. A phishing attack could be used as an example. These are thoroughly planned criminal attacks, where websites are designed to masquerade as the authentic site, in order to obtain another individual's authentication credentials and confidential information illegally for financial gain. Phishing attacks are mostly distributed over e-mail as this is one of the easier ways to reach a large distribution of the population in order to ensure the success of the attack. A disparity is created between the individual's perception and the truth, eliciting a heightened fear response, where cognitive abilities are compromised, and the probability of ensuring that the correspondence is legitimate will be minimal [9].

Overloading: This technique has a time element, with the result that the individual becomes cognitively pacified or compliant, through the bombardment of a series of hurried persuasive axioms [8].

Reciprocation: "One good deed deserves another"; Social exchange theory states that individuals, on receiving a kind gesture from another, feels obligated to reciprocate with kindness. The social engineer might create a problem for the individual, only to fix it again, in order to make the individual feel obligated to reciprocate by disclosing information [7].

Deceptive Relationship: To obtain information, the social engineer will identify an individual to purposefully build and establish a relationship. This is done with a particular purpose, as individuals tend to share information freely within established relationships [8].

Diffusion of responsibility and moral duty: The individual is made to believe that their actions - to disclose information, even though it is against policy - will have greater benefits and important beneficial consequences, such as to help save an employee or helping the institution, and that they will not be held solely responsible for their actions [8].

Authority: By the social engineer portraying an authority figure, the individual is more likely to comply with the request to disclose information, as an authority figure almost implicitly elicit a conditioned response to adhere to their wishes and demands, combined with a fear of punishment if the individual may appear to undermine their authority by verifying their legitimacy [7],[8].

Integrity and Consistency: Individuals have an intrinsic desire to uphold their commitments, even if it were not their own [8].

These triggers could be used to perform a social engineering attack on an unsuspecting victim, which could lead the victim to experience a sense of discomfort, whether just an uneasiness or even anxiety, as all these attacks prey on the victim's psychological vulnerabilities. One would expect that a victim would be able to use these clues of discomfort to detect that he is being targeted by a social engineering attack. However, this is the ideal and not reality, as the human reasoning and decision-making process is extremely complex, and prone to error.

The following section discusses the human reasoning and decision-making process and how it applies to detecting social engineering attacks.

III. HUMAN REASONING

The human ability to make conscious, rational judgments, which underlie their decisions, will not always be the ideal. This can be ascribed to various human factors, such as limited information-processing capacity, the use of heuristics (mental processes, or shortcuts, used to simplify the process of judgment, which can lead to judgmental error), personal preferences, and a vulnerability to be influenced by emotions and manipulated by others. Human decision making is a complex process, where most decisions that need to be made will not have only one ideal option, and the same decision will not be made by all people [10],[11].

Within the subjective utility theory, the subjective experience of an individual is taken into consideration, where the goal is to maximize gain and to avoid losses. This subjective experience refers to the individual's own personal judgment on value (utility) and likelihood (probability), instead of objective criteria and computations, where personal characteristics have an impact [11],[12].

The individual will follow a series of steps to come to a decision. First, for each option, they will multiply the subjective probability by the positive subjective utility, followed by subtracting the calculation, as before, for negative subjective utility. Based on these expected values, individuals will make their decision [11].

Risk will always be an integral part of decision-making, as the possible outcome is uncertain. The subjective expected utility theory is the most widely applied model regarding risk decisions. Within this extended version of the subjective expected utility theory, it allows for subjective probabilities, where judgments are made based on the person's belief on likelihood, and where no objective mathematical probabilities are available. This theory cannot, however, predict human

decisions. As indicated by the term subjective, each person will have their own set of values and characteristics. By considering the particular individual's subjective expected utilities and their subjective estimates of probabilities of cost and benefits, one can predict the optimal decision for that particular individual [10],[11].

Within this subjective expected utility theory model it is believed that the individual will try to achieve a well-reasoned decision by considering all the possible alternatives and information available, calculating the probability of each probable outcome and the cost and benefits it may hold [10]. Based on this theory, a decision to disclose information will be based on risk-benefit analysis [10].

Decision analysis, a technology based on subjective expected utility theory, attempts to aid better decision making [10]. This approach attempts to aid people to comprehend and have clarity regarding their goals and values, to search for possible options and verification of facts. One of the techniques used by decision analysis is decision trees. Decision trees are representations of decisions, which aid complex decision-making by breaking it down into more manageable components. Values are assigned to each element, whereupon ideal decision principles are applied to integrate these elements. By combing the probabilities and the utilities that correspond to each possible outcome, the best alternative is selected [10].

People do not possess a stable set of pre-existing values that are simply applied; their decisions will be determined by the present context, and the demands of the decision [10].

As indicated by literature, individuals find it difficult to make rational decisions in a limited time frame, especially regarding complex matters. With the skill of the social engineer and the complexity of the attack he is performing, at best, an uninformed individual would only be able to make an educated guess regarding the likelihood of being targeted by a social engineering attack. An individual would need a predefined set of guidelines on which to measure the likelihood of a social engineering attack in order to make a more informed decision.

The following section is devoted to proposing a practical application model to determine if a social engineering attack is being performed.

IV. SOCIAL ENGINEERING ATTACK DETECTION MODEL (SEADM)

As indicated, a model is needed as guideline to detect social engineering attacks. The authors propose a social engineering attack detection model, making use of a decision tree, by breaking the process down into more manageable components, and guidelines to aid decision-making (SEADM) in figure 1.

This paper firstly addresses each of these states individually as shown in figure 1 before the full model is discussed with examples. Throughout this discussion the term individual is

defined as the person dealing with the incoming call and the term requester is defined as the person whom is making the call and requesting the information.

A. How would you describe your emotional state?

The first necessary step in this model, and one that will have to be considered throughout the process, would be for the individual to be conscious of, and able to evaluate their emotional state, on an ongoing basis. This implies a consciousness of emotion and how it can affect one's decisions.

In the same manner, the individual should evaluate the emotion the requester elicit within themselves, as the psychological vulnerabilities, that might be triggered by a social engineering attack, is directly aimed to create certain emotional states in order to obtain information.

We are all familiar with a day that start off horribly and seem to continue with every possible thing going wrong. For example, the car broke down on the way to work, followed by a negative emotional experience whether it be family problems or a argument with a spouse or colleague. All factors and negative events influence our emotional state and hamper our ability to make rational, thought-through decisions [13]. In such a negative emotional state it is more likely to be a victim of social engineering: concentration is low, irritability and frustration is high, where an individual can possibly provide a requester with information just to get rid of them.

It is necessary to emphasize again what a critical role an individual's emotional state can play in the safekeeping of privileged information. If an individual is in a negative emotional space, the individual will not always be able to make a rational decision on the sensitivity level of the information of a request, or to whom it may be disclosed. This could result in costly losses to the institution and individual.

Awareness and consciousness of one's emotional state will not be an easy task, or even always a possible task for all individuals. With training and rehearsal this skill can and will improve. For this reason the authors are in the process of developing a quick self-evaluation electronic questionnaire that individuals will be able to use. However, in combination with the model, training by the institution can be emphasized on the various techniques used, the psychological vulnerabilities the attacker may elicit, and institutional policy and procedures.

It is important to note that judging one's own emotional state could be a tedious matter and some individuals are unable to perform this task. It is for this reason that an automated self-evaluation electronic questionnaire would be implemented. The questionnaire would consist of a large database of questions, of which only a few would be used per evaluation of this state. Only a few would be used each time as there is a time constraint associated with the model and individuals would be unable and reluctant to perform a self-evaluation task if it takes an excessive amount of time. The timeframe for completing this state should be within a few seconds.

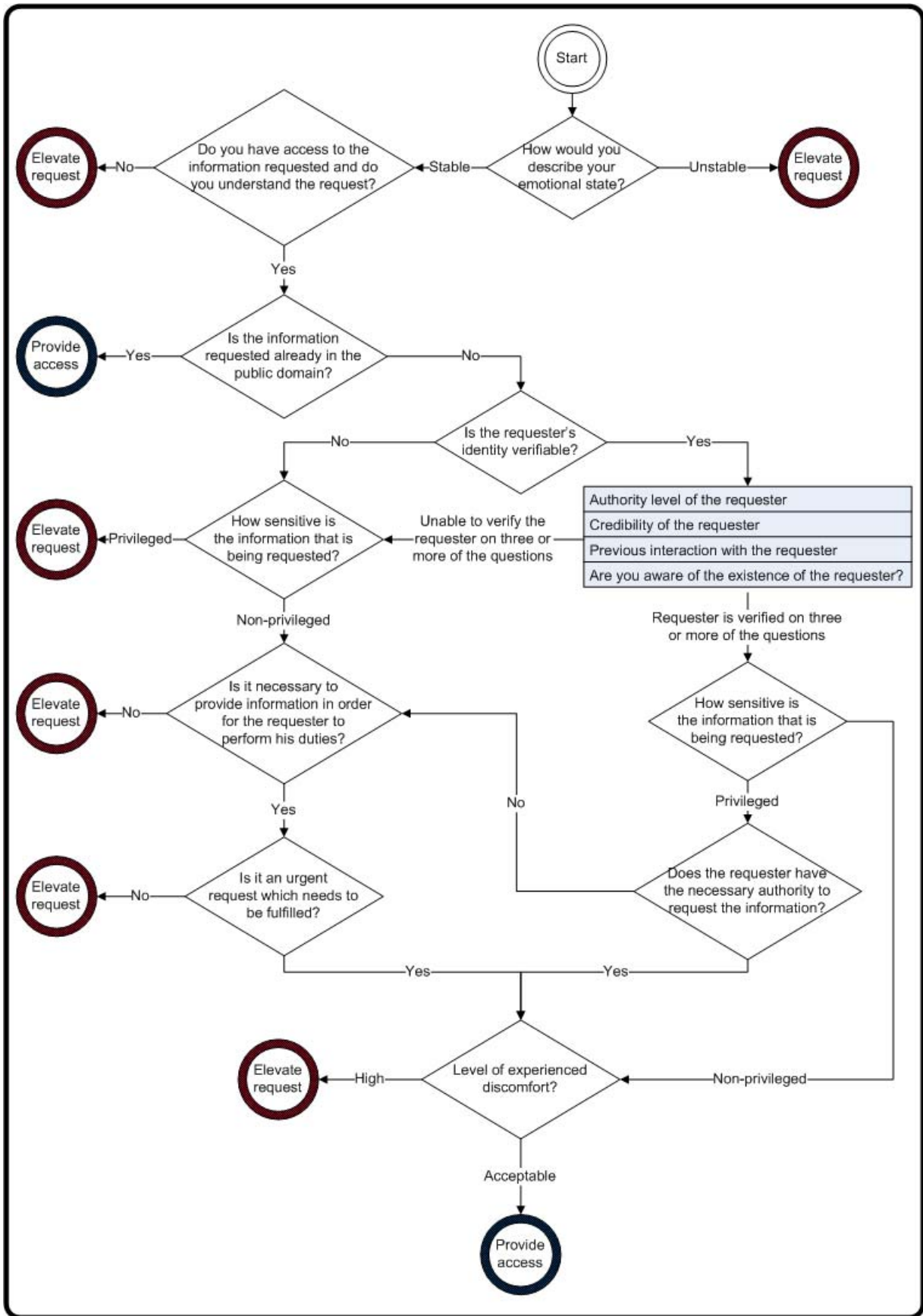


Figure 1. Social Engineering Detection Model

If the individual or the self-evaluating questionnaire finds that the individual is too emotional, the call or email request should rather be escalated to another individual. Of course this has the implication, and danger, of people using this as a tool to shift their work responsibilities to another, as well as promote further frustration for all people involved. However, the dangers of social engineering, obtaining privileged information, which can lead to great losses to the institution and possibly the individual, are a much greater threat.

B. Do you have access to the information requested and do you understand the request?

When a request for particular information is made, the individual needs to judge if they possess adequate knowledge regarding the requested information, and if they have access to the information which is being requested, to adequately provide this information. Obviously, if the individual does not have the knowledge required, they will not be able to provide the information, and could refer the requester to another individual, who will also then follow this model. If the individual judge that they have adequate knowledge on the subject in question the following step can be taken.

C. Is the information requested already in the public domain?

Individuals should have a clear understanding of what information are readily accessible to the public regarding their institution and related information. The information in the public domain could include contact details and working hours, which could be available on the institutions website, and thus be legally provided to a requester.

D. Is the requester's identity verifiable?

The individual now needs to verify the identity of the requester, to enable them to make an informed and rational decision if information should be provided to the requester at a later stage of the model. If the requester's identity cannot be verified, a different set of states will be examined to determine if the information should be provided.

Important to remember is that the social engineer might be portraying himself as an authority figure within the institution, a computer technician, or any other persona that might elicit compliance. As humans we are inclined to make quick assumptions regarding people and their stature, based on trivialities such as clothing. If someone is dressed in the proper attire, use the appropriate institutional jargon, using an important individual's name, does not necessarily indicate that the individual is trustworthy. Social engineers do an enormous amount of research before an attack if warranted. If at any time the individual feels unsure, they should contact their manager, to obtain authority to provide, or not provide, the information requested.

To verify the requester's identity, the following should be taken into account and used to form a global impression to base the decision on whether to provide or not provide the requested information: authority, credibility, previous interaction, and knowledge of the person's existence will have to be taken into account.

Some of the techniques that can aid in the verification process of an individual's identity are the following: Caller Identification; Calling back the requestor on a predetermined phone number; To request a secure email; To request a secure password; To request a face-to-face interaction with the individual, where he would provide proper identification, Where another employee can vouch for the requester; To contact the requester's immediate supervisor in order to verify his/her identity; To use an employee directory [3].

In this model it is suggested that the individual should be able to determine at least three of the four components to successfully verify the individual. Each of these qualities will now be individually addressed.

1) Authority level of the requester

Authority is part of any institution, with an almost conditioned response from employees to adhere to their wishes and demands, combined with a fear of punishment if the individual may appear to undermine their authority [7]. For this reason it is a very effective technique used by social engineers to obtain privileged information. The institution needs to provide an environment where the employee feels comfortable, and are expected to question the authority figure's identity when disclosing sensitive information.

With the determining authority, the employee also needs to know, with the help of a clear institutional policy, what authorization level a particular person of authority has, in regards to what privileged information can be provided.

2) Credibility of the requester

The employee needs to judge the level of credibility of the requester. However, this is a challenging task, as establishing credibility is the first step the social engineer undertakes, and what the attack will be based on.

If the requester knows the jargon used by the particular institution, people easily assume that the requester is an employee at their particular institution. The requester could, for example, be an ex-employee, quite knowledgeable about the jargon and procedures. Such ex-employee might seek revenge with the goal of obtaining particular sensitive information.

The credibility of the requester is measured on the basis of how he/she responds on predefined of set of questions which can be used to determine the credibility of a requester.

3) Previous interaction with the requester

If the individual had previous interaction with the requester, especially a longstanding history of interaction, the decision and knowledge to what information can be provided will be an easy task. However, few interactions with the requester, especially by telephone and email alone, should be considered in conjunction with other verification techniques, to be able to make an informed and safe decision regarding the disclosure of information.

4) Are you aware of the existence of the requester?

This refers to the knowledge that the requester exists within the institution or an outside collaborating partner on a project can support the verification of the requester. However, this should also be used in conjunction with the other verification techniques, as the requester could be a social engineer

portraying himself as the well-known figure in order to obtain privileged information.

It is suggested that within institutional policies and procedures, a classification system of information should be established, whereupon a document should be compiled and made available to all, of all personnel indicating what level of information authorization each has, which will simplify the process.

E. How sensitive is the information that is being requested?

It is critical that the individual are knowledgeable, and have absolute clarity, what information is privileged, and what information are authorized to be provided, and to whom, thus depicting the level of information sensitivity. This skill can be nurtured and enhanced through training on institutional policies and procedures.

For the purpose of this model, information is divided into two categories, privileged and non-privileged information. Privileged information indicating information requiring a form of authorization, and non-privileged information indicating information that requires no authorization and are freely available.

The proposed model should be used in conjunction with an institution's policies and procedures on information sensitivity. These policies and procedures should include clear, easily understandable and easily accessible guidelines to verify the authorization level needed in order to request the specific information.

As each institution is unique, each will have to create and establish their own security policies to address the classification of sensitivity of particular information, under which circumstances it may be divulged, and to which particular individuals or institutions. These policies should also include processes and accountability for reporting suspected incidents [7].

After determining whether information is privileged or non-privileged, the individual will need to determine if the requester has the necessary authority to request the information.

F. Does the requester have the necessary authority to request the information?

With the aid of the previous steps the individual possess the necessary knowledge regarding the requester's identity and authority level, together with the information classification. The individual can now determine whether the requester has a level of authority on the same level or higher as the level of sensitivity of the information. If the requester possess authorization on the same authority level or higher needed for the particular information, the next step - the level of experienced discomfort - can be considered.

However, if the authority figure does not have the necessary authorization, or if the individual feels that the request made is not legitimate, the model will treat the requester as a non-verified individual. In this scenario the following step - to determine the necessity of the information to fulfill required duties - will be considered.

G. Is it necessary to provide information in order for the requester to perform his duties?

A subjective estimation needs to be made if it will be beneficial or detrimental to provide the information to the requester at the particular time of the request, as well as how it could empower the individual to complete their work. The individual should be sure that if he/she provides information to the requester that it would indeed be beneficial to both parties involved.

Apart from establishing if the information would help the requester to complete his duties, one would also need to consider the urgency of the request.

H. Is it an urgent request which needs to be fulfilled?

The individual needs to assess the urgency that the requested information is needed. If the information is not urgently needed, and any doubts exist, the information does not have to be provided, or can be provided at a later time. With the time leniency, an authority can be consulted, who can choose to further investigate, or provide authorization to divulge the requested information.

If the information is urgently needed, whether it is to complete an urgent project, or in a life threatening situation such as where an individual's medical insurance number is required as he was injured at work, the employee should consider the next step of level of experienced discomfort.

I. Level of Experienced Discomfort

Evaluation of one's emotions is again emphasized, where an individual will have to trust the emotions they are experiencing at that particular time, e.g. "trust your gut". If the level of discomfort experienced is evaluated as too high, information should rather not be provided, as certain techniques used by social engineers may elicit high levels of emotional discomfort, enabling them to obtain privileged information. Part of the social engineer's skill set is the ability to profile individuals, using the appropriate technique for the particular individual, forcing them into a desired role. This technique is called altercasting [1]. In a certain scenario they may be aggressive and threatening towards the individual, causing high levels of anxiety, where the individual's cognitive ability to reason, to be able to stay calm and focused, and to be able to make rational counterarguments, are detrimentally influenced. In another scenario, and also the most frequently used form of this technique, the individual will be ascribed to the role of helper, where the individual could experience discomfort and possibly guilt - an emotion most people try to avoid - if they do not oblige to the request.

If, however, the individual does not experience any discomfort or if the level of discomfort is understandable and acceptable, information can be provided, as the previous steps have been successfully completed.

The next section demonstrates the application of the model by use of examples.

V. DISCUSSION

Three example scenarios are provided within this section. The first scenario is a legitimate request by a bank account

holder, requesting his bank account balance. The second scenario also depicts a request for a bank account balance, however, by a social engineer. The third depicts a basic scenario where a request is made regarding the closing time of a store.

Within all the provided scenarios in this paper, it will be assumed that the individual dealing with the request is in a stable emotional state.

A. Scenario one

A telephonic request is made to obtain a personal bank account balance. The process, according to the SEADM model, will be following:

Emotional state of the call centre agent will be analyzed, which will equate to stable.

Do you have access to the information requested and do you understand the request? Yes.

Is the information requested already in the public domain? An individual's bank balance is not public information and will, thus, be necessary for the agent to verify the identity of the requestor.

The requestor will need to identify himself, and establish his credibility by providing the call centre agent with his personal information. The call centre agent then verifies the information by comparing it to the information on the system when the bank account was created. This verifies the question of being aware of the existence of this requestor, as well as the authority level of the requestor.

How sensitive is the information being requested? A bank account balance is classified as privileged information.

Does the requester have the necessary authority to request the information? Yes.

Lastly the call centre agent would need to analyze his level of experienced discomfort, which would be acceptable as there were no issues in this call.

Within the process completed, in this scenario, access can be provided, allowing the call centre agent to provide the requestor with his bank balance.

B. Scenario two

This scenario also depicts a request for a bank account balance, however, by a social engineer.

Emotional state will be analyzed, which will equate to stable.

Do you have access to the information requested and do you understand the request? Yes.

Is the information requested already in the public domain? An individual's bank balance is not public information, and will thus be necessary for the agent to verify the identity of the requestor.

The requestor, who, in this scenario is a social engineer, will attempt to identify himself. This can proceed in one of two ways. The social engineer could be in possession of adequate information pertaining to the victim's personal and banking

details. This information used in conjunction with his various skills and techniques, for example overloading, can convince the call centre agent he is the legitimate requester. This could lead the call centre agent to experience a high level of discomfort. The call center agent could elevate the request to another individual with higher authority to adequately manage the request, or could deny access to the information.

To fully explain the model, this paper will examine the alternative route, where the social engineer failed to validate himself as the owner of the bank account but he has validated himself as a friend of the owner of the bank account.

How sensitive is the information being requested? A bank account balance is classified as privileged information.

Does the requester have the necessary authority to request the information? Within this alternative scenario the answer would be no. A friend will not have authorization to privileged information as a bank account balance.

Is it necessary to provide information in order for the requester to perform his duties? The social engineer could portray himself as the bank account holder's accountant, explaining that he needs the information to complete his duties. Assuming the call centre agent allows this, he will move onto the urgency test.

The call centre agent needs to determine the urgency of the request. However, a legitimate accountant would ask the account holder to contact the bank and obtain the necessary information. In this scenario the call centre agent would need to elevate the request, and report the request as a suspicious.

This scenario depicts how a social engineering attack could have been thwarted. The last scenario depicts a request to public information.

C. Scenario three

Within this scenario a request is made regarding the closing time of the institution.

Emotional state will be analyzed, which will equate to stable.

Does the individual have access to the information requested and understand the request? Within this scenario the individual have the necessary information regarding the operating hours of the institution and understands what information is being requested.

The operating hours of the institution is information which is already in the public domain, and thus can be provided to the requester.

This paper concludes by providing a brief summary and the potential advantages it may hold to an institution if applied together with adequate training.

VI. CONCLUSION

Social engineering is very difficult to detect, as the social engineer possess various skills and effective techniques, preying on human vulnerabilities, which makes these attacks often go without notice. What makes detection even more difficult is that many people are unaware of this technique and

the potential threat, and dire consequences it holds for the individual and for institutions.

As of yet, only training has predominantly been considered as preventative measure to social engineering. However, it has been shown that training is soon forgotten, especially in the real work environment, rendering training only as ineffective against social engineering. It is proposed that a visible practically applied, user-friendly aid, as the SEADM, will aid in the daily awareness of the threat, and thus protection against social engineering.

It has been shown by the use of scenarios that the proposed model is indeed feasible as a preventative measure to social engineering attacks. This model makes a valuable contribution to the field of social engineering, as it aids in the detection of social engineering attacks, by breaking down the decision-making process into manageable components.

Future research will aim to improve the SEADM, by designing an automated electronic emotional self-evaluation questionnaire. This will aid the model by removing the subjectivity from the emotional state question and provide an objective way to measure an individual's emotional state. The authors will also explore research by [2] to illustrate the probable increase in awareness of an individual's own vulnerability to such an attack, through practical application of social engineering in training. Lastly, some action research in a call centre will be completed in order to verify the usability of SEADM.

REFERENCES

[1] K D Mitnick and William L Simon, *The art of deception: controlling the human element of security*. Indianapolis: Wiley Publishing, 2002.

- [2] J W Scheeres, R F Mills, and M R Grimaila, "Establishing the human firewall: reducing an individual's vulnerability to social engineering attacks," in *3rd International Conference on Information Warfare and Security*, April 2008.
- [3] K D Mitnick and William L Simon, *The art of intrusion: the real stories behind the exploits of hackers, intruders and deceivers*. Indianapolis: Wiley Publishing, 2005.
- [4] J Debrosse and D Harley, "Malice through the looking glass," in *Virus Bulletin Conference*, September 2009.
- [5] G L Orgill, G W Romney, M G Bailey, and P M Orgill, "The urgency for effective user privacy-education to counter social engineering attacks on secure computing systems," in *Conference On Information Technology Education*, Salt Lake City, October 2004, pp. 177-181.
- [6] Gragg D. (2002, December) A Multi-Layer Defense Against Social Engineering. Sans Institute Reading Room.
- [7] M Workman, "A test of interventions for security threats from social engineering," *Information Management & Computer Security*, vol. 16, no. 5, pp. 463-483, 2008.
- [8] A Chandler and R Boadhurst, "Social Engineering and Crime Prevention in Cyberspace," Queensland University of Technology, Brisbane, 2006.
- [9] X Dong, J A Clark, and J L Jacob, "User behaviour phishing websites detection.," *Proceedings of the IMCSIT*, vol. 3, pp. 783-790, 2008.
- [10] N Braisby and A Gellatly, *Cognivite Psychology*.: Oxford University Press, 2005.
- [11] R J Stemberg, *Cognitive Psychology*, 4th ed.: Thomson Watsworth, 2006.
- [12] G Bansal, F M Zahedi, and D Gefen, "The impact of personal dispositions on information sensitivity, privacy concern and trust in disclosing health information online," *Decision Support Systems*, vol. 49, no. 2, pp. 138-150, May 2010.
- [13] M T Siponen, "A conceptual foundation for organizational information security awareness," *Information Management & Computer Security*, vol. 8, no. 1, pp. 31-41, 2008.

Requirements for wireless sensor networks in order to achieve digital forensic readiness

F Mouton and HS Venter

Information and Computer Security Architecture, University of Pretoria
moutonf@gmail.com

Abstract

The field of wireless sensor networking is a new and upcoming one and unfortunately still lacking as far as digital forensics is concerned. All communications between different nodes (also known as *motest*) are sent out in a broadcast fashion. These broadcasts make it quite difficult to capture data packets forensically whilst retaining their integrity and authenticity. This paper examines the differences between IEEE 802.15.4 wireless sensor networks and IEEE 802.11x wireless networks when it comes to implementing digital forensic readiness within the network environment. It focuses on the differences in the communication protocol, proof of authenticity and integrity, time stamping, modification of the network after deployment and other differences between IEEE 802.15.4 wireless sensor networks and IEEE 802.11x wireless networks. Each of these elements is discussed, after which a table is provided that shows the specific requirements to be taken into account when proposing digital forensic readiness in a wireless sensor network environment.

Keywords

forensic readiness, digital forensics, wireless sensor networks

1. Introduction

Our pursuit of a better lifestyle has led to a vast improvement in the technology to which we have access in today's world. The concept of a wireless sensor network (WSN) is just another technology developed to improve our ability to better accomplish our daily tasks. The implementation of security protocols on WSNs has not received much attention to date, and, even more so, very little consideration has been given to digital forensics within a WSN environment.

The problem is that currently there is no formal set of requirements for achieving digital forensic readiness in wireless sensor networks. The purpose of this paper is to determine how IEEE 802.15.4 wireless sensor networks differ from IEEE 802.11x wireless networks when it comes to implementing digital forensic readiness.

The remainder of the paper is structured as follows: The second section provides some background information about WSNs and digital forensic readiness. Section 3 discusses the differences between IEEE 802.11x wireless networks and IEEE 802.15.4 wireless sensor networks with regard to digital forensic readiness. Section 4 proposes a set of requirements that need to be adhered to when implementing digital forensic readiness for wireless sensor networks. Finally, a summary is provided of the forensic readiness requirements that are proposed and of future work to be done in this field.

2. Background

Wireless sensor networks still constitute a relatively new area of research in computer science and the first papers on WSNs were only published in the last decade (Chong & Kumar, 2003;

Mouton & Venter, 2009). Much of the research on WSNs has been dedicated to new areas of application aimed at supporting our modern lifestyle. Some background information for a better understanding of WSNs is provided next, before strategies for the achievement of digital forensic readiness for WSNs are suggested.

2.1 Wireless Sensor Networks

WSNs belong to the general family of sensor networks that use multiple distributed sensors to retrieve data from various environments of interest. Chong and Kumar (2003) provide a history of previous accomplishments of WSNs and show how they have evolved in terms of sensing, communication and computing. WSNs consist of wireless nodes with embedded processors and ad hoc networks (Estrin et al., 2001), and involve wireless communication (Ye, Heidemann & Estrin, 2002). Mouton and Venter (2009) define a WSN as an ad hoc network that consists of tiny and resilient computing nodes known as *moten* or sensors. These *moten* are extremely efficient with regard to power consumption and can collaborate effectively with other *moten* in their vicinity. A graphical representation of a wireless sensor network is provided in Figure 1, while in Table 1 the functions of each of the components are subsequently summarised briefly (Mouton & Venter, 2009; Heinzelman, Kulik & Balakrishnan, 1999; Sohrabi et al., 2000).

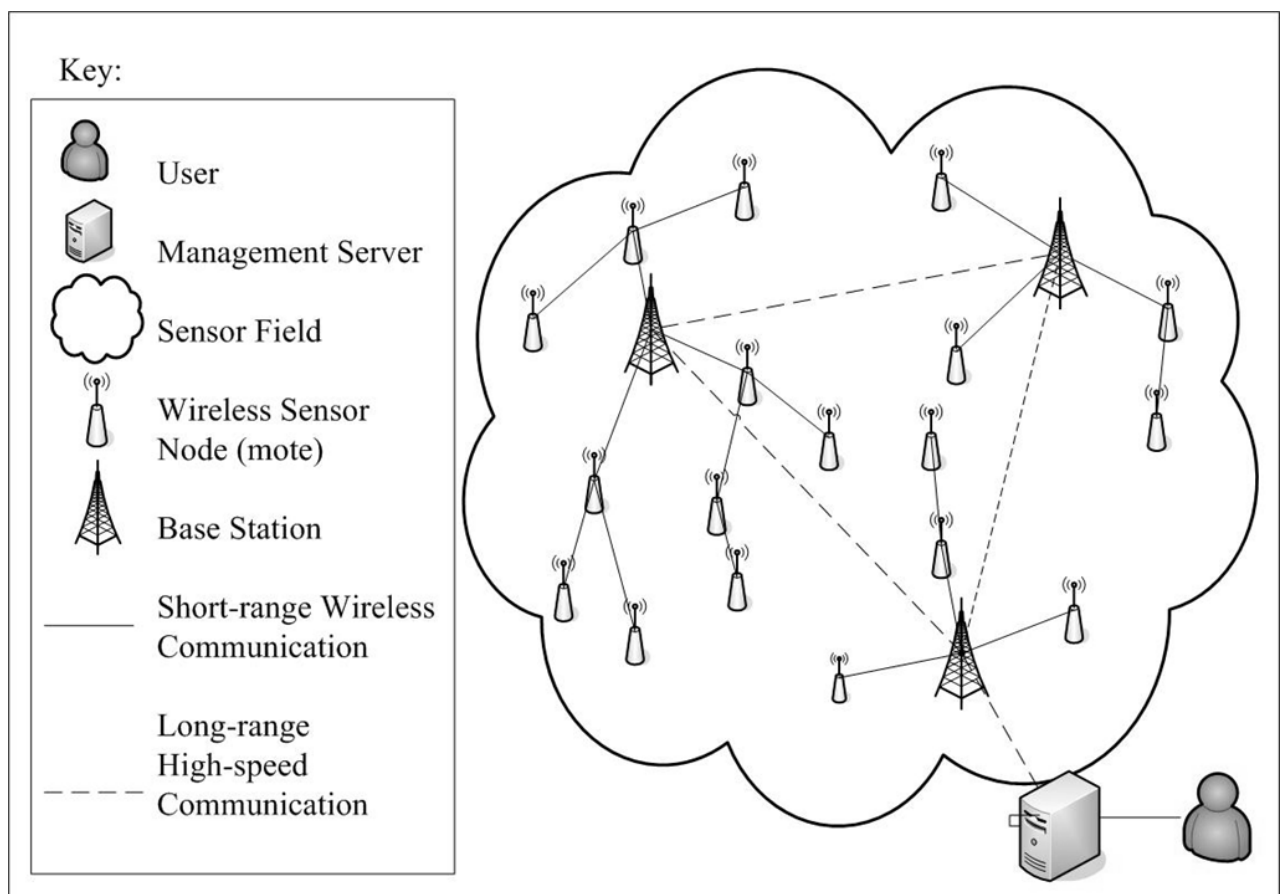


Figure 1: A graphical representation of a wireless sensor network (Mouton & Venter, 2009).

WSN component	Functions of each component
User	The user can interact with the WSN through the management server.
Management Server	The management server serves as an interface console for the WSN.
Sensor Field	The sensor field denotes the physical boundaries of the WSN.
Wireless Sensor Node (<i>mote</i>)	Each <i>mote</i> contains a small subset of the various sensors. <i>Motes</i> in the network can also act as repeaters for packets that need to reach the base station.
Base Station	A base station serves as a gateway node through which the information of the <i>motes</i> has to travel to reach the management server.
Short-range Wireless Communication	Short-range wireless communication links are established between neighbouring <i>motes</i> and the neighbouring base stations.
Long-range High-speed Communication	Long-range high-speed communication links are established between further-ranged base stations and the management server.

Table 1: Brief summary of functions of the components of a wireless sensor network (Mouton & Venter, 2009).

WSNs can be used in many environments. Their *motes* may consist of many different types of sensors, such as thermal, visual, infrared, radar or acoustic. These *motes* can monitor a wide variety of ambient conditions, including humidity, pressure, sound, noise levels, temperature, lightning conditions and objects moving through a designated area (Elson & Estrin, 2001; Kahn, Katz & Pister, 1999).

Some applications of WSNs include military applications such as the tracking of moving objects and battlefield surveillance (Zhao, Shin & Reich, 2002); environmental applications such as habitat monitoring, forest fire detection and flood detection (Mainwaring et al., 2002); and health applications such as the tracking and monitoring of doctors and patients in hospitals, as well as of drug administration in hospitals (Lu et al., 2002). Finally, WSNs can also be used for home and building automation applications.

The next subsection focuses on providing the reader with a workable definition of digital forensic readiness in a WSN context.

2.2 Digital Forensic Readiness

To achieve digital forensic readiness in any type of environment, it is essential to first establish an acceptable definition for it. However, since it is a fairly new concept and the subject of divergent opinions, consensus must still be reached in this regard.

In defining digital forensic readiness, Tan (2001) identifies two objectives that have to be balanced carefully: maximising the ability to collect credible digital evidence, and minimising the cost of performing a digital forensic investigation. Tan also argues that several steps need to be taken to ensure that an environment is ready as far as digital forensics is concerned. Rowlingson (2004), on the other hand, suggests ten steps that describe the key activities in implementing a digital forensic readiness programme. Because Rowlingson's steps have actually been designed to create a business process model for digital forensic readiness, this paper gives preference to Tan's objectives for meeting the requirements of digital forensic readiness in a WSN environment.

Even though Tan's two objectives provide a very good definition of digital forensic readiness, it is important to refine them somewhat to make the definition more specific to a WSN environment. For the purpose of this paper, digital forensic readiness is defined as the notion to perform a digital forensic investigation in the shortest amount of time with the least amount

of cost and without having to disrupt the original network that has to perform mission-critical tasks. This definition is set as the main goal for achieving digital forensic readiness on WSNs.

The next section discusses the differences between IEEE 802.15.4 wireless sensor networks and IEEE 802.11x wireless networks when it comes to implementing digital forensic readiness.

3. Differences between WSNs and WLANs

WSNs have special needs compared to IEEE 802.11x wireless networks and hence have more specialised requirements than would apply to wireless networks (also known as wireless local area networks or WLANs). There are many important factors that make a WSN unique and distinguish it from a WLAN. The factors that are addressed in this paper are the following:

- Communication protocol
- Proof of authenticity and integrity
- Time stamping
- Modification of the network after deployment
- Protocol data packets
- Radio frequencies
- Power supply
- Network overhead
- Data integrity

The factors listed above are the main ones that differentiate WSN environments from WLAN environments. The reasoning behind the choice of these factors will become apparent in the coming subsections, where each factor is addressed individually. It is, however, important to remember that the core of the argument about the importance of these factors concerns the manner in which they influence the design decision of how to implement a digital forensic readiness application for WSNs.

While examining each of these factors, it is important to note that the authors assume that no modification to the original WSN (hence forward referred to as *oWSN*) is allowed and thus a secondary independent forensic WSN (hence forward referred to as *fWSN*) would be used for the digital forensic readiness implementation of the *oWSN*.

The discussions in each subsection below briefly focus on how these factors differ from WLAN to WSN, and subsequently our focus shifts to how to address them in WSNs.

3.1. Communication Protocol

All communication within a WSN occurs in a broadcast fashion and thus a *mote* never really knows which of its neighbouring *notes* actually receives the packet (Akyildiz et al., 2002; Tseng, Ni & Shih, 2003). The default functioning of a *mote* in the sensor field is to receive all packets – upon receipt of a packet it then has to analyse if the packet was meant for it or not. This analysis requires some processing that drains the battery of the *mote*, which is an important consideration in WSN communication.

The broadcasting technique used in WSNs is very different from the communication techniques used in an IEEE 802.11x wireless network. In the WLAN environment, one can determine if a packet has arrived at its destination by monitoring the network, since

acknowledgement packets are sent to confirm the receipt of packets (Xylomenos & Polyzos, 1999; Xylomenos et al., 2001). This is not the case in a WSN environment.

Due to the broadcasting fashion in which WSNs communicate, the *mote* that broadcasts packets will never be completely sure whether the packet was received by the *mote* for which the packet was intended. This uncertainty could be overcome by introducing a communication protocol that allows the receiving *mote* to reply with a receipt acknowledgement packet. However, because this would require extra transmissions that can lead to a greater battery drain, this procedure cannot simply be implemented in all WSNs. The suggested technique also has several other disadvantages. If a flooding attack is launched against the *oWSN*, it would compel the *oWSN* to reply to each flooding attempt with receipt acknowledgement messages, which would then flood the entire *oWSN*.

Considering that a protocol founded on receipt acknowledgement packets can have such a severe impact on a WSN environment, it seems quite impractical to use such a protocol in this environment. Hence the authors have agreed to accept that most WSN *notes* will be uncertain as to whether or not packets have actually arrived at their destination. This causes severe problems in terms of forensic monitoring with a secondary network. It could likely be the case that the packets received by the *oWSN* base station might differ from those received by the *fWSN* base station in the case that some of the packets are dropped in either of the two WSNs. In the case of the *fWSN*, however, this problem could be avoided by implementing a protocol that uses receipt acknowledgement packets, because it is in the nature of a forensic network to always be sure that the information received at either point of the communication line contains some degree of authenticity and integrity. In order to achieve sound digital forensic readiness, it is crucial to prove the authenticity and integrity of the data packets that have been received. The next subsection focuses on defining what the authors see as authenticity and integrity. The differences between maintaining the authenticity and integrity from a WLAN and a WSN perspective are also discussed, as well as possible ways of maintaining authenticity and integrity within a WSN environment.

3.2. Proof of Authenticity and Integrity

Authenticity and integrity first need to be defined as there could be different opinions on precisely what each of them means. In the context of this paper, authenticity is defined as the certainty that the origin and destination of the data packet are kept intact throughout its whole lifetime. The lifetime of a data packet runs from the time that it is sent from the first *mote* up to the time when it is received and processed by the base station. Next, integrity is defined as the certainty that the correctness of the data within the data packet is kept intact throughout the lifetime of the data packet.

Numerous techniques for proving the authenticity and integrity of packets in an IEEE 802.11x wireless network have already been published (Chen, Jiang & Liu, 2005; Komori & Saito, 2004; (Guizani & Raju, 2005). Firewalls, Intrusion Detection Systems, Wireless Routers and Wireless Network Interface Cards are all examples of equipment you would find in an IEEE 802.11x wireless network and most of these devices have the ability to generate a log or some other way of showing which data packets have passed through the network. Most of these abilities are fairly simple techniques that are performed by the device itself. In most cases where a log file is generated, it is safe to assume that the information reflected in the log file is actually the true pattern of traffic that has passed through the device. However, this is only the case when it is certain that the device is not defective or that the log file has not been tampered with. This single log file can also be backed up by looking at all the other devices

through which this single packet has travelled, as most devices in an IEEE 802.11x environment should have some form of logging. In a WSN environment, however, very little or no logging is done on the *motes* in the sensor field, due to various reasons. These reasons can include the limited power source and the limited storage space that these devices have. WSN equipment, by default, only does logging at the base station and if logging were to be required at every *mote*, one would have to go and implement this yourself. This obviously raises another issue, namely as to the trustworthiness of the code with which one does the logging. Tried and tested techniques for logging are generally more trustworthy than one's own attempts at implementing logging. It is easier to defend the authenticity and integrity of a well-known logging technique than that of a self-developed technique. In the case where a self-developed technique is used, it must be based on some solid theory as to why it can provide authenticity and integrity. Because WSNs differ so significantly from WLANs, the authors have decided to propose a form of logging that is based on the Casey Certainty Scale (Casey, 2002).

Fortunately, in a WSN environment, multiple *motes* tend to be able to each capture the same data packet simply because they are all in range of a particular broadcasted packet. This is a feature of WSNs, which is not the case in IEEE 802.11x networks. Most devices in WLANs will ignore packets that are not meant for them and do not even attempt to log these packets. The opposite is true for WSNs, where *motes* attempt to capture every data packet within range. This feature of WSNs can be successfully exploited in an attempt to prove the authenticity and integrity of packets in the WSN. All the packets captured by each independent *fWSN mote* could be forwarded to the base station, as a central point of analysis, in an effort to prove the authenticity and integrity of the data packet according to the Casey Certainty Scale (Casey, 2002).

According to Casey (2002), the integrity and authenticity of information is more certain if this information was recorded by different independent sources. Each *mote* can, in essence, be seen as an independent source. Thus, the authenticity and integrity of each packet can be determined based on the number of *motes* in the network that have received the same broadcasted packet. This paper therefore assumes that, in accordance with the Casey Certainty Scale (Casey, 2002), a packet that has been seen by a larger number of *motes* has far greater authenticity and integrity than a packet that has only been seen by a few forensic *motes* in the network.

The above technique constitutes only one of several ways to determine the authenticity and integrity of the packets in a WSN. Time stamping and the sequence of packets can also be used for this purpose. However, time stamping in a WSN is a tedious task. The next subsection is nevertheless devoted to it.

3.3. Time stamping

Time stamping in a WLAN environment is a fairly easy task, since all the devices in a WLAN would under normal conditions either have access to a time server or have been set with the correct time. Thus, time stamping in the logs for a WLAN would under most conditions be correct, provided that the device has not been tampered with or is not faulty. In the case of a WSN, however, only the management server (which is connected to the base station) has a sense of time. The *motes* in a WSN environment have no sense of physical world time and the only measurement they can use is their own sense of time, which is the time that has elapsed since they were switched on (Sundararaman, Buy & Kshemkalyani, 2005; Su & Akyildiz, 2005; Sun, Ning & Wang, 2006). Such elapsed time can be measured on WSN devices in

terms of ticks, where each tick represents 100 nanoseconds (Sundararaman, Buy & Kshemkalyani, 2005). This uptime, although fairly accurate, is a poor indication of time, because each *mote* in the entire network has to be switched on simultaneously and the time should also be synchronised by transmitting their uptime along with their data packets. It is impractical to switch on *motest* simultaneously and synchronisation is not feasible due to resource restrictions.

When tests were conducted concerning the time stamping of WSNs, the authors noted that it takes at most one second to capture any data packet and transmit it to the *fWSN* base station. This nevertheless introduced a time delay between capturing a packet and receiving it at the base station. The time delay also differed according to the distance of the *fWSN mote* from the base station in terms of hops and physical distance. Thus the time stamps at the base station are not an accurate reflection of when the packet was initially captured, as the base station is the only device that can assign an accurate time stamp if it is connected to the management server. (The reason for this is that only the management server has access to a time server (Sundararaman, Buy & Kshemkalyani, 2005; Su & Akyildiz, 2005).) It is also important to note that each *fWSN mote* captures packets sequentially, in the order that the *oWSN motest* transmit their data packets. This proves to be a vital piece of information, because one would then be able to claim that even if the time stamps are altered, the sequence would still be intact. The order in which they arrive at the *fWSN* will stay intact even if the time stamps are slightly delayed. This allows one to assume that the time delay between capturing the packet and sending it to the forensic base station would not really affect the authenticity and integrity of the packets, as the sequence of packets can be used to determine their authenticity and integrity.

The trustworthiness of log time stamps is an issue that many digital forensics researchers have queried and investigated (Schatz, Mohay & Clark, 2006; Schneier & Kelsey, 1999). The dilemma faced by the *fWSN* is merely intensified. It becomes a more severe issue to trust the time stamps as the limitation as having no access to a centralised time server for WSNs might prevent them from reflecting the correct time. However, since the sequence of the data packets is not altered, this (rather than the time stamps) could be used to verify the authenticity and integrity of the data packets. This paper therefore assumes that the fixed sequence of the data packets is more important than the precise time at which they were transmitted. More information can be gathered by looking at the sequence of the data packets than by looking at their time of transmission.

It is therefore sufficient to capture the data packets and merely provide a time stamp for them as soon as they arrive at the *fWSN* base station. In the event that this is done, one would admittedly create a time stamp error. The time stamp error would nonetheless be a constant error for each *oWSN mote* respectively, as it would reflect the time the data packet was first transmitted together with the added time it took for this data packet to reach the *fWSN* base station. The *fWSN* base station, which is connected to a time server, assigns a time stamp to each data packet upon its arrival there. This allows the order of the packets to be kept intact and records a one-second error on the time stamp of each packet due to the fact that the base station assigns the time stamps and not the forensic *mote* that captured the packet initially. The time stamp error stays constant for all the packets received from a specific *mote* in the sensor field and thus it is still possible to guarantee the authenticity and integrity of a packet. This constant error could be measured, if needed, by comparing the time stamps at the *oWSN* base station and the *fWSN* base station. The time stamp, combined with the sequence of the data packets, would then be sufficient to be used in a forensic investigation.

Another issue that the authors have considered while examining the differences between WLANs and WSNs is the feasibility of modifying the network after it has been deployed. This matter is discussed in the following subsection.

3.4. Modification of the network after deployment

Being able to modify the network after deployment is the only factor that was found to be fairly similar between WLANs and WSNs, as it is always possible to modify code on a device by retracting it from the field, redeveloping it and then redeploying it back into the field. However, the practicality of altering *oWSN* devices after deployment must be taken into consideration. It is important to remember that *oWSN* *motes* are usually scattered in an area and to alter them, one would have to go and collect the entire network and redeploy it. Hence, it seems essential that the *oWSN* should not be modified to accommodate an *fWSN* solution. This is the very reason why the authors have opted to add an overlaying *fWSN* to the *oWSN* in order to do all the forensic monitoring. The overlaying *fWSN* would consist of a separate set of WSN *motes* that does not affect the *oWSN* and also requires no modification of the *oWSN*.

The difficulty and impracticality of modifying the *oWSN* has led the authors to believe that this should also be seen as a specific requirement when attempting to provide forensic readiness to a WSN environment. Considering that we cannot easily alter the *oWSN*, we must ensure that the *fWSN* should be able to handle any type of protocol headers and footers that could originate from the *oWSN*. Against this background, the next subsection focuses on the protocol data packets that are used by WSN devices and the reasons why it is important to take this into consideration when implementing an overlaying *fWSN*.

3.5. Protocol Data Packets

The *oWSN* can have many different types of communication protocols in its normal operation. For example, the data packets can include packets to determine the routing protocol, sensory packets, encrypted packets or even malformed packets. In order to ensure that all of the possible protocols used in WSNs are encapsulated in this approach, it has been assumed that the *oWSN* uses an address-free protocol. This protocol generates the largest amount of network overhead in WSNs, as it would cause data to be sent from a source *mote* in the network to every other *mote* in the network on each data transmission (Dunkels, Osterlind & Zhitao, 2007). The most commonly used address-free protocols are data dissemination protocols, where neither the sender *mote* nor any of the other *motes* in the network knows the address of the receiving *mote*. If the *fWSN* is able to successfully log this communication of an address-free protocol in a way that ensures authenticity and integrity, one could assume that the name-based WSN protocols would effortlessly be accounted for, as they have much less network overhead (Dunkels, Osterlind & Zhitao, 2007).

As is also the case in WLANs, the *motes* in the *fWSN* should listen in promiscuous mode and should be able to handle any type of packet that is transmitted or received by the *oWSN*. The authors define promiscuous mode to be a configuration of the WSN *mote* in which all traffic within the WSN *mote*'s frequency range and wireless range will be received by the WSN *mote*. Thus, if an attacker uses a foreign *mote* to inject data into the *oWSN*, the *fWSN* should also be able to listen in on this data. This requirement should be fairly simple to adhere to, because if the *fWSN* is implemented on the same type of equipment, it should be possible to intercept all communication.

Lastly, the *fWSN* should be using a name-based WSN protocol for communication between other *fWSN* *motes* as it is more optimal in terms of network overhead than address-free protocols. In name-based protocols the source *mote* knows the address of the receiving *mote* and the *motes* between the sender and receiver know the path to the receiving *mote* (Dunkels, Osterlind & Zhitao, 2007).

All the major differences between WSNs and WLANs have now been discussed. Due to space constraints, discussions on radio frequencies, power supply, network overhead and data integrity have been omitted. However, the following section is devoted to arranging all these factors, including the ones that have been excluded from the discussion, into a single workable list of requirements that need to be adhered to when implementing digital forensic readiness in a WSN environment.

4. Forensic Readiness Requirements for WSNs

The previous sections identified the factors that differentiate between WLANs and WSNs in terms of digital forensic readiness. These factors were simply broad overviews of issues to be considered in the WSN environment (most of which do not exist in a WLAN environment).

The authors consequently propose a broad, yet detailed set of the important requirements to be adhered to in order to successfully implement digital forensic readiness in a WSN environment. This list of requirements (see Table 2) could serve as a good starting point for anyone working on digital forensic readiness and makes it easier for an individual to implement digital forensic readiness within a WSN environment. Most other researchers focus mainly on one or two of these requirements by going into more detail on them in their research papers, but many other requirements are usually not mentioned, regardless of their importance.

Table 2 therefore gives a quick but comprehensive overview and summarises *all* the important requirements that need to be taken into account in order to achieve digital forensic readiness in an IEEE 802.15.4 WSN environment.

Factors	Detailed requirement list
Communication Protocol	1. The <i>fWSN</i> should use a receipt acknowledgement packet protocol to ensure that all data packets captured by the <i>notes</i> in the field do indeed reach the base station.
	2. The broadcasted communication from the <i>oWSN</i> should be intercepted in a manner that ensures that the data packets are not altered in any fashion.
	3. The <i>fWSN</i> should be able to capture all possible types of communication that can be sent from the <i>oWSN</i> .
Proof of Authenticity and Integrity	4. The authenticity and integrity of all the data packets should remain intact while being captured on the <i>fWSN</i> .
	5. The data packets that are captured in the <i>fWSN</i> should be stored in such a way that their authenticity and integrity are not compromised.
	6. It should be possible to verify the authenticity and integrity of all the data packets in case a digital investigation takes place.
Time Stamping	7. The data packets should have a time stamp assigned to them that does not violate their authenticity and integrity.
	8. The sequence of the packets captured should reflect the true sequence in which they were transmitted from the original network.
Modification of the network after deployment	9. It should be possible to implement the <i>fWSN</i> without any modification of the <i>oWSN</i> .
Protocol Data Packets	10. The <i>fWSN</i> should be designed in such a manner that the network topology or the routing protocol used by the <i>oWSN</i> does not influence the <i>fWSN</i> 's operation.
Radio Frequencies	11. The <i>fWSN</i> should be able to communicate on the same radio frequencies that are available to the <i>oWSN</i> .
	12. All communication within the <i>fWSN</i> should occur on a frequency not utilised in the <i>oWSN</i> .
	13. If an intruder WSN is in the area and communicates on a frequency that influences the <i>oWSN</i> , then the <i>fWSN</i> should be able to forensically capture these data packets.
Power Supply	14. The <i>fWSN</i> should not increase power consumption in the <i>oWSN</i> and the <i>fWSN</i> should have at least the same or a longer network lifetime than the <i>oWSN</i> in terms of battery power.
Network Overhead	15. While intercepting communication, there should be no extra network overhead on the <i>oWSN</i> .
Data Integrity	16. The <i>fWSN</i> should by no means be able to influence the <i>oWSN</i> or influence any sensory data transmitted within the <i>oWSN</i> .

Table 2: Requirements in order to achieve digital forensic readiness in a IEEE 802.15.4 WSN environment

The list in table 2 provides a sound basis to start from when attempting to achieve digital forensic readiness in a WSN environment. The following section concludes this paper and proposes future work.

5. Conclusion

Wireless sensor networks constitute a type of network that makes any type of digital forensic analysis very difficult due to the nature of the network. This paper therefore proposed a list of requirements that need to be taken into consideration when implementing digital forensic readiness for an IEEE 802.15.4 wireless sensor network.

The main aim of this paper was to establish the differences between IEEE 802.15.4 wireless sensor networks and IEEE 802.11x wireless networks from a digital forensic readiness point of view. The problem was that currently there is no formal set of requirements for successfully implementing digital forensic readiness in wireless sensor networks. This problem was addressed by focusing on the special needs WSNs have for digital forensic readiness and providing a list of requirements that need to be taken into account when implementing digital forensic readiness in WSNs.

In future research, the authors intend to explore this list of requirements in greater detail and develop a digital forensic readiness prototype for wireless sensor networks. The focus of the research will be to develop the prototype in such a way that it proves to be robust enough to function in most types of WSNs.

6. References

- Akyildiz, I.F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) 'Wireless sensor networks: a survey', *Computer Networks*, vol. 38, no. 4, pp. 393-422.
- Casey, E. (2002) 'Error, Uncertainty and Loss in Digital Evidence', *International Journal of Digital Evidence*, vol. 1, no. 2, Summer.
- Chen, J., Jiang, M. and Liu, Y. (2005) 'Wireless LAN security and IEEE 802.11i', *Wireless Communications, IEEE*, vol. 12, no. 1, February, pp. 27-36.
- Chong, C. and Kumar, S.P. (2003) 'Sensor networks: evolution, opportunities, and challenges', *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247-1256.
- Crossbow Technology Inc (2007) *Imote2 Hardware Reference Manual*, Revision A edition, San Jose: Crossbow Technology Inc.
- Doufexi, A., Armour, S., Butler, M., Nix, A., Bull, D., McGeehan, J. and Karlsson, P. (2002) 'A comparison of the HIPERLAN/2 and IEEE 802.11a wireless LAN standards', *IEEE Communications Magazine*, vol. 40, no. 5, May, pp. 172-180.
- Dunkels, A., Osterlind, F. and Zhitao, H. (2007) 'An adaptive communication architecture for wireless sensor networks', Proceedings of the 5th international conference on Embedded networked sensor systems, Sydney, Australia, 335-349.
- Elson, J. and Estrin, D. (2001) 'Time synchronization for wireless sensor networks', Proceedings of the 15th International Symposium on Parallel and Distributed Processing, 1965-1970.
- Estrin, D., Girod, L., Pottie, G. and Srivastava, M. (2001) 'Instrumenting the world with wireless sensor networks', Proceedings of the 2001 IEEE International Conference on Acoustics, Speech and Signal Processing, 2033-2036.
- Guizani, M. and Raju, A. (2005) 'Wireless Networks and Communications Security', in Xiao, Y., Li, J. and Pan, Y. (ed.) *Security and Routing in Wireless Networks*, 3rd edition, New York: Nova Science Publishers.
- Heinzelman, W.R., Kulik, J. and Balakrishnan, H. (1999) 'Adaptive protocols for information dissemination in wireless sensor networks', In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, 174-185.
- Kahn, J.M., Katz, R.H. and Pister, K.S. (1999) 'Next century challenges: mobile networking for "Smart Dust"', In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, New York, 271-278.
- Komori, T. and Saito, T. (2004) 'A secure wireless LAN system retaining privacy', 18th International Conference on Advanced Information Networking and Applications, Kanagawa, 370-375.
- Lu, C., Blum, B.M., Abdelzaher, T.F., Stankovic, J.A. and He, T. (2002) 'RAP: a real-time communication architecture for large-scale wireless sensor networks', Proceedings of the 8th IEEE International Workshop on Real-Time and Embedded Technology and Applications Symposium, 55-66.
- Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R. and Anderson, J. (2002) 'Wireless sensor networks for habitat monitoring', In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, New York, 88-97.
- Mouton, F. and Venter, H.S. (2009) 'A Secure Communication Protocol for Wireless Sensor Networks', Proceedings of the Annual Security Conference "Security Assurance and Privacy: organizational challenges", Las Vegas.
- Polastre, J., Hill, J. and Culler, D. (2004) 'Versatile low power media access for wireless sensor networks', Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, 95-107.
- Rowlingson, R. (2004) 'A Ten Step Process for Forensic Readiness', *International Journal of Digital Evidence*, vol. 2, no. 3.
- Schatz, B., Mohay, G. and Clark, A. (2006) 'A correlation method for establishing provenance of timestamps in digital evidence', Proceedings of the 6th Annual Digital Forensics Research Workshop, 98-107.

- Schneier, B. and Kelsey, J. (1999) 'Secure audit logs to support computer forensics', *ACM Transactions Information System Security*, vol. 2, no. 2, May, pp. 159-176.
- Shnayder, V., Hempstead, M., Chen, B., Allen, G.W. and Welsh, M. (2004) 'Simulating the power consumption of large-scale sensor network applications', In Proceedings of the 2nd international Conference on Embedded Network Sensor Systems, Baltimore, 188-200.
- Slijepcevic, S. and Potkonjak, M. (2001) 'Power efficient organization of wireless sensor networks', In IEEE International Conference on Communications, 472-476.
- Sohrabi, K., Gao, J., Ailawadhi, V. and Pottie, G.J. (2000) 'Protocols for self-organization of a wireless sensor network', *Personal Communications, IEEE Wireless communications*, vol. 7, no. 5, October, pp. 16-27.
- Su, W. and Akyildiz, I.F. (2005) 'Time-diffusion synchronization protocol for wireless sensor networks', *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 384-397.
- Sundararaman, B., Buy, U. and Kshemkalyani, A.D. (2005) 'Clock synchronization for wireless sensor networks: a survey', *Ad Hoc Networks*, vol. 3, no. 3, pp. 281-323.
- Sun, K., Ning, P. and Wang, C. (2006) 'TinySerSync: secure and resilient time synchronization in wireless sensor networks', Proceedings of the 13th ACM conference on Computer and communications security, Alexandria, 264-277.
- Tan, J. (2001) *Forensic Readiness*, Technical Report edition, Cambridge: @Stake.
- Tseng, Y., Ni, S. and Shih, E. (2003) 'Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network', *IEEE Transactions on Computers*, vol. 52, no. 5, pp. 545-557.
- Wander, A.S., Gura, N., Eberle, H., Gupta, V. and Shantz, S.C. (2005) 'Energy analysis of public-key cryptography for wireless sensor networks', Third IEEE International Conference on Pervasive Computing and Communications, 324-328.
- Xylomenos, G. and Polyzos, G. (1999) 'TCP and UDP Performance over a Wireless LAN', In Proceedings of the IEEE INFOCOM.
- Xylomenos, G., Polyzos, G., Mahonen, P. and Saaranen, M. (2001) 'TCP performance issues over wireless links', *IEEE Communications Magazine*, vol. 39, no. 4, pp. 52-58.
- Ye, W., Heidemann, J. and Estrin, D. (2002) 'An energy-efficient MAC Protocol for wireless sensor networks', Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communication Societies, 1567-1576.
- Zhao, F., Shin, J. and Reich, J. (2002) 'Information-driven dynamic sensor collaboration for tracking applications', *IEEE Signal Processing Magazine*, vol. 19, pp. 61-72.

A prototype for achieving digital forensic readiness on wireless sensor networks

Francois Mouton

Department of Computer Science
Information and Computer Security Architecture
Pretoria, South Africa
moutonf@gmail.com

H.S Venter

Department of Computer Science
Information and Computer Security Architecture
Pretoria, South Africa
hventer@cs.up.ac.za

Abstract—The field of wireless sensor networking is still a new and upcoming one and, unfortunately, still lacking in terms of digital forensics. All communications between different nodes (also known as motes) are sent out in a broadcast fashion. These broadcasts make it quite difficult to capture data packets forensically whilst retaining integrity and authenticity of the data packets. This paper examines whether and how one can add a digital forensic readiness layer to an existing IEEE 802.15.4 wireless sensor network without any modification to the existing wireless sensor network. This paper also provides demonstrations of a working prototype to show that a digital forensic readiness layer can be added to an existing wireless sensor network, if the prototype adheres to a list of requirements in order to achieve digital forensic readiness in a wireless sensor network environment. This is done by performing several demonstrations which resemble real world wireless sensor network scenarios in order to show that the prototype does indeed add a layer of digital forensic readiness to the existing wireless sensor network.

Keywords- forensic readiness, digital forensic; wireless sensor networks

I. INTRODUCTION

Striving towards a better lifestyle has led to a great improvement in the technology we have access to in today's world. The concept of a wireless sensor network is just another technology developed to improve our ability to accomplish our daily tasks. The implementation of security protocols on WSNs has not received much attention up to date, and, even more so, very little focus has been shed on digital forensics within a WSN environment. The motivation for this paper is the need to explore and expand on the field of digital forensic readiness, focusing on wireless sensor networks. The authors feel that there is still much research required in this area.

The problem is that currently there exists only a list of requirements in order to achieve digital forensic readiness in a wireless sensor network, and these requirements have not yet been tested by means of a prototype and with real world wireless sensor network scenarios. The goal of this paper is, therefore, to test if these requirements are sufficient in order to achieve digital forensic readiness in a wireless sensor network. The prototype, specifically, attempts to demonstrate whether a digital forensics layer can be added on top of an existing IEEE 802.15.4 wireless sensor network without any modification to

the existing IEEE 802.15.4 wireless sensor network. The purpose of this paper is to design a prototype according to the list of requirements and perform demonstrations in order to show the usability of the list of requirements. This paper also shows that it is possible to implement digital forensic readiness on an existing wireless sensor network without any modification to the existing wireless sensor network.

The remainder of the paper is structured as follows: the second section provides a background to WSNs and digital forensic readiness. Section three provides the requirements for achieving digital forensic readiness in IEEE 802.15.4 wireless sensor networks. Section four demonstrates the prototype which has been implemented taking all the requirements from section three into account. Finally, the last section concludes with an overview of the demonstrations and proposes future work.

II. BACKGROUND

First, Wireless Sensor Networks (WSNs) still comprise a relatively new area of research in computer science and the first papers on WSNs only appeared around the start of the 21st century [1], [2]. Much of the research on WSNs has been on new areas of application aimed at supporting our modern lifestyle. Some background information for a better understanding of WSNs is provided next before a solution to the digital forensic readiness for WSNs is suggested.

A. Wireless Sensor Networks

WSNs belong to the general family of sensor networks that use multiple distributed sensors to retrieve data from various environments of interest. Chong and Kumar [1] provide a history on previous accomplishments of WSNs and show how they have evolved in terms of sensing, communication and computing. WSNs consist of wireless nodes with embedded processors and ad hoc networks [3], and involve wireless communication [4]. Mouton and Venter [2] define a WSN as an ad hoc network that consists of tiny and resilient computing nodes known as motes or sensors. These motes are extremely efficient with regard to power consumption and can collaborate effectively with other motes within their vicinity. A graphical representation of a wireless sensor network is provided in Figure 1 and the functions of each of the components are briefly summarised in Table 1 [2], [5], [6].

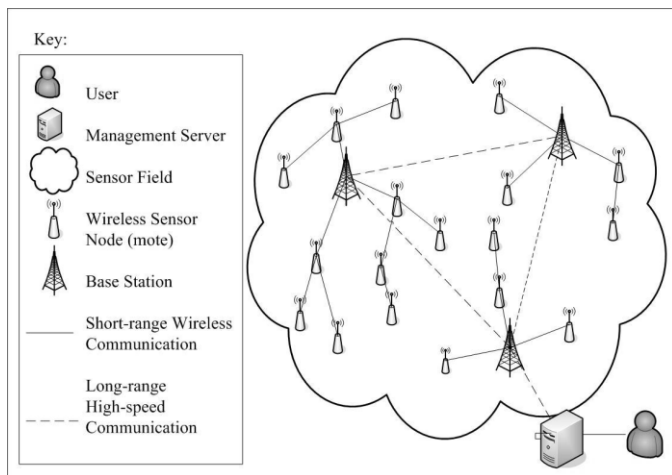


Figure 1 A graphical representation of a wireless sensor network [2].

TABLE I. BRIEF SUMMARY OF FUNCTIONS OF THE COMPONENTS OF A WIRELESS SENSOR NETWORK [2]

WSN component	Brief summary of the component
User	The user can interact with the WSN through the management server.
Management Server	The management server serves as an interface console for the WSN.
Sensor Field	The sensor field denotes the physical boundaries of the WSN.
Wireless Sensor Node (mote)	Each mote contains a small subset of the various sensors. Motes in the network can also act as repeaters for packets that need to reach the base station.
Base Station	A base station serves as a gateway node through which the information of the motes has to travel to reach the management server.
Short-range Wireless Communication	Short-range wireless communication links are established between neighbouring motes and the neighbouring base stations.
Long-range High-speed Communication	Long-range high-speed communication links are established between further-ranged base stations and the management server.

WSNs can be used in many environments. Their motes may consist of many different types of sensors, such as thermal, visual, infrared, radar or acoustic. These motes can monitor a wide variety of ambient conditions, including humidity, pressure, sound, noise levels, temperature, lightning conditions and objects moving through a designated area [7], [8].

Some applications of WSNs include military applications such as the tracking of moving objects and battlefield surveillance[9]. Environmental applications include habitat

monitoring, forest fire detection and flood detection [10]. Health applications include the tracking and monitoring of doctors and patients in hospitals, as well as drug administration in hospitals [11]. Finally, WSNs can also be used for home and building automation applications.

The next subsection focuses on providing the reader with a workable definition for digital forensic readiness in a WSN context.

B. Digital Forensic Readiness

To achieve digital forensic readiness in any type of environment, it is essential to establish an acceptable definition for it. However, since it is still a fairly new concept, many people have different opinions about it.

Tan [12] identifies two objectives as part of a definition for digital forensic readiness that have to be carefully balanced: maximising the ability to collect credible digital evidence, as against minimising the cost of performing a digital forensic investigation. Tan also argues that several steps need to be taken to ensure that an environment is ready as far as digital forensics is concerned. On the other hand Rowlingson [13] suggests ten steps that describe the key activities in implementing a digital forensic readiness programme. Because Rowlingson's steps have actually been designed to create a business process model for digital forensic readiness, this dissertation gives preference to Tan's two objectives for meeting the requirements of digital forensic readiness in a WSN environment.

Even though Tan's objectives provide a very good definition of digital forensic readiness, it is still important to refine this to make it more specific to a WSN environment. For the purpose of this paper, digital forensic readiness is defined as the notion to perform a digital forensic investigation in the shortest amount of time with the least amount of cost and without having to disrupt the original network that has to perform mission critical tasks. This definition is set as the main goal for achieving digital forensic readiness on WSNs.

The definition provided focuses on three elements. These three elements are, respectively: the time period required to perform a digital forensic investigation, the cost involved in performing a digital forensic investigation and the ability to collect the evidence without disrupting the environment.

Each of these three elements is discussed separately in the following subsections as they are pivotal to the achievement of digital forensic readiness.

1) Time Period Required to Perform a Digital Forensic Investigation

Digital forensic readiness is put in place in order to decrease the time period it takes to perform a digital forensic investigation [12].

In an environment where digital forensic readiness is implemented, the time it takes from when the incident occurs until the incident-related information can be analysed is kept to a minimum. This is because the digital forensic readiness systems ensure that the information is captured into a separate

environment on which the work-related systems are not dependent [13].

The nature of the digital forensic readiness environment being separate from the work-related environment brings us to the next important factor: the cost involved in performing a digital forensic investigation.

2) Cost Involved in Performing a Digital Forensic Investigation

In the case where a digital forensic investigation is required in an environment which is not compliant to a digital forensic readiness solution, this could potentially cost the organisation a large amount of money [12].

Considering the cost involved in implementing digital forensic readiness versus the cost involved in conducting a digital forensic investigation without having digital forensic readiness implemented, it would be fair to say that one would rather be safe than sorry. In other words, one would rather incur the costs of digital forensic readiness and be ready should an incident occur, than have to pay more money should an incident occur [12], [14].

This subsection briefly touched on the cost involved in digital forensic readiness, the next subsection focuses on how digital forensic readiness can also be used to minimise costs by preventing disruption of the environment.

3) Collecting Evidence without Disrupting the Environment

The collection of evidence in a digital forensic readiness environment would occur far more easily than in an environment without digital forensic readiness. This is due to the nature of the digital forensic readiness environment, whereby it is separated from the mission critical environment.

One would be able to take down an entire digital forensic readiness system if it has been configured to be running as a separate system from the main environment. This allows one to thoroughly perform the collection of digital evidence without the added pressure of time limitations.

Considering the time saved, cost minimized and the lack of disruption caused to the environment by the implementation of a digital forensic readiness environment, this paper has shown how an organisation would benefit from the implementation of such an environment. The next section very briefly shows the requirements in order to achieve digital forensic readiness in a wireless sensor network environment.

III. REQUIREMENTS IN ORDER TO ACHIEVE DIGITAL FORENSIC READINESS

In a previous paper by Mouton and Venter [15], it has been shown which requirements are needed for a prototype in order to achieve digital forensic readiness in a wireless sensor network environment. The table from the previous paper is provided here as it is an important reference point to the prototype in order to implement digital forensic readiness on wireless sensor networks.

In the table, and also for the rest of the paper, the term ‘original WSN’ is referred to as oWSN while the term ‘secondary independent forensic WSN’ is referred to as fWSN. The oWSN is the existing WSN which is already deployed in the field, whereas the fWSN is the digital forensic readiness network which adds a digital forensic readiness layer on top of the oWSN.

TABLE II. REQUIREMENTS IN ORDER TO ACHIEVE DIGITAL FORENSIC READINESS IN AN IEEE 802.15.4 WSN ENVIRONMENT [15]

Requirements in order to achieve digital forensic readiness in an IEEE 802.15.4 WSN environment
1. The fWSN should use a receipt acknowledgement packet protocol to ensure that all data packets captured by the motes in the field do indeed reach the base station.
2. The broadcasted communication from the oWSN should be intercepted in a manner which ensures that the data packets are not altered in any fashion.
3. The fWSN should be able to capture all possible types of communication which can be sent from the oWSN.
4. The authenticity and integrity of all the data packets should remain intact whilst they are being captured on the fWSN.
5. The data packets which are captured in the fWSN should be stored in such a way that the authenticity and integrity is not compromised.
6. The authenticity and integrity of all the data packets should be able to be verified in case a digital investigation takes place.
7. The data packets should have a timestamp assigned to them that does not violate their authenticity and integrity.
8. This sequence of the packets captured should reflect the true sequence in which they were transmitted from the original network.
9. The fWSN should be able to be implemented without any modification of the oWSN.
10. The fWSN should be designed in such a manner that the network topology or the routing protocol used by the oWSN does not influence the fWSN’s operation.
11. The fWSN should be able to communicate on the same radio frequencies as the ones which the oWSN is capable of using.
12. All communication within the fWSN should occur on a frequency which is not utilised in the oWSN.
13. If an intruder WSN is in the area and communicates on a frequency which influences the oWSN then the fWSN should be able to forensically capture these data packets.
14. The fWSN should not increase the power consumption in the oWSN and the fWSN should have at least the same network lifetime or longer than the oWSN in terms of battery power.
15. Whilst intercepting communication there should be no extra network overhead on the oWSN.
16. The fWSN should by no means be able to influence the oWSN or influence any sensory data transmitted within the oWSN.

Having provided this list of requirements, the following section will demonstrate a working protocol, which takes into

account all these requirements in order to achieve digital forensic readiness in a WSN environment.

IV. DIGITAL FORENSIC READINESS PROTOTYPE

The proposed digital forensic readiness prototype adheres to all the requirements which have been set and presented in the previous section. The implementation of the prototype was done on the Crossbow Imote2 boards. The Crossbow Imote2 is an advanced WSN node platform designed for demanding WSN applications that require high processing power wireless link performance and reliability [16]. We do realise that the Crossbow Imote2 is an extremely powerful mote. This was, however, the only mote at our disposal in the development of the prototype. The Crossbow Imote2 motes can communicate on the frequency between 2.405 GHz and 2.480 GHz with each channel separated by 5 MHz [16]. All the coding for the implementation was done in .net Micro Framework Version 2.0 as it is supported by the Crossbow Imote2 circuit boards [16].

Three demonstrations were done to show that it is possible to provide digital forensic readiness whilst adhering to the constraints which were provided in the previous section. For the first demonstration, a sample WSN network was used as the original network. The original network was able to measure temperature, light and humidity. In the first demonstration the network was setup as can be seen in figure 2.

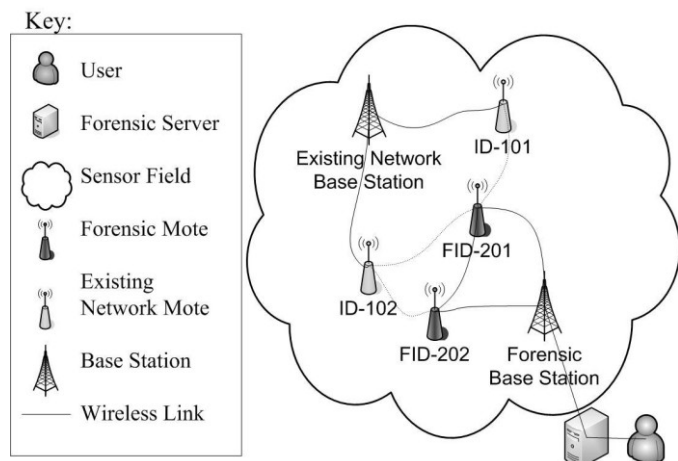


Figure 2 A graphical representation of the network layout for the first demonstration.

In figure 2 each mote has a unique MoteID allocated to it as specified in the figure. The wireless link is used to show which motes are in range of a specific mote to receive its broadcasted messages. The original mote, labelled ID-102, can be seen by both digital forensic WSN motes FID-202 and FID-201, whereas original mote ID-101 can only be seen by FID-201. This should cause a resulting log where all the packets transmitted from ID-102 should be seen by both FID-201 and FID202, however, all the packets transmitted by ID-101 are only seen by FID-201. The small scale of the demonstrations is due to the limited size of the network as the authors only had access to four WSN motes and two base stations for testing purposes.

In figure 3 it can be seen how the prototype displays the log file which has been generated from the digital forensic base

station. The columns have all been provided with names to make it easier to recognise which data is being displayed in the figure. Figure 5, however, is included to show that all the data is actually only recorded in terms of bytes and that the column names are actually labeled “Byte 1” through to “Byte 64”.

Line	Date	Seen By	Forensic MoteID	Originating MoteID	Byte 6	Light Value
1	12/07/2009 12:23:07:562	201	201	101	6	22
2	12/07/2009 12:23:17:750	201	201	101	7	21
3	12/07/2009 12:23:18:203	201, 202	201	102	4	54
4	12/07/2009 12:23:38:234	201	201	101	8	21
5	12/07/2009 12:23:38:843	202, 201	202	102	5	53
6	12/07/2009 12:23:48:539	201	201	101	9	21
7	12/07/2009 12:23:58:828	201	201	101	10	21
8	12/07/2009 12:23:59:562	202, 201	202	102	6	52
9	12/07/2009 12:24:09:148	201	201	101	11	22
10	12/07/2009 12:24:19:437	201	201	101	12	22
11	12/07/2009 12:24:20:250	202, 201	202	102	7	53
12	12/07/2009 12:24:29:718	201	201	101	13	21
13	12/07/2009 12:24:40:031	201	201	101	14	21
14	12/07/2009 12:24:40:750	202, 201	202	102	8	54
15	12/07/2009 12:24:50:328	201	201	101	15	21
16	12/07/2009 12:25:00:625	201	201	101	16	22
17	12/07/2009 12:25:01:296	202, 201	202	102	9	53
18	12/07/2009 12:25:10:937	201	201	101	17	21
19	12/07/2009 12:25:21:234	201	201	101	18	21
20	12/07/2009 12:25:22:046	202, 201	202	102	10	52
21	12/07/2009 12:25:31:531	201	201	101	19	21
22	12/07/2009 12:25:41:000	201	201	101	20	22

Figure 3 Demonstration to show how packets on the outskirts of the network are handled

It is important to note that there are some columns omitted on the snapshots of the demonstration figures. This was done to retain readability of the snapshots and only the information which applies to each demonstration was kept on each snapshot. In figure 3 the “Line” column is simply a line number indicator which makes it easier to discuss specific lines from the figure in this paper. The grey colour in the “Line” column is added if the data packet was seen by more than one mote in the FWSN. The “Date” column represents the timestamp for the packet. The “Seen By” column represents which motes in the FWSN were in range of the packet when it was transmitted. The “Forensic MoteID” column represents the digital forensic mote in the FWSN which was the closest to the packet and thus received it first. The “Originating MoteID” column represents which mote in the OWSN transmitted this packet. The “Byte 6” column is the packet number which is transmitted by the motes in the OWSN and this increases by one for each transmission. The “Light Value” column is a light measurement which is one of the purposes of the original network – to measure light intensity. The motes in the OWSN also measures other things like humidity and temperature, but we only use the light intensity sensor for the purposes of this paper as this is sufficient enough to provide proof of concept. It can be seen from figure 3 that FID-202 was only able to see communication coming from ID-102 as indicated by the “Originating MoteID” column, whereas FID-201 was able to see communication from both ID-101 and ID-102. Thus, if this information were to be used in a digital forensic analysis, one would be more certain of the authenticity and integrity of the data packets which have been sent by ID-102 as both digital forensic motes were able to see communication coming from it. It can, however, be said that the information received from ID-101 also has a degree of authenticity and integrity because the data which was received from the FWSN can be matched with the data from the OWSN. The authenticity and integrity of ID-101’s data can also be verified by realising that the data packets that ID-101 sent were only seen by FID-201, as indicated by the “Seen By” column, and this is consistent throughout the

entire log. The data received from the FWSN could also be used to verify data from the OWSN if there happens to be a dispute about the legitimacy of the data. Figure 4 shows the data which was received in the same demonstration session as figure 3 but here the data is sorted according to which OWSN mote has sent the data.

Line	Date	Seen By	Forensic MoteID	Originating MoteID	Byte 6	Light Value
28	12/07/2009 12:26:23:031	201	201	101	24	21
30	12/07/2009 12:26:33:328	201	201	101	25	21
31	12/07/2009 12:26:43:625	201	201	101	26	21
33	12/07/2009 12:26:53:921	201	201	101	27	21
34	12/07/2009 12:27:04:234	201	201	101	28	21
36	12/07/2009 12:27:14:531	201	201	101	29	21
37	12/07/2009 12:27:25:125	201	201	101	30	21
39	12/07/2009 12:27:35:453	201	201	101	31	21
40	12/07/2009 12:27:45:718	201	201	101	32	21
3	12/07/2009 12:23:18:203	201, 202	201	102	4	54
5	12/07/2009 12:23:38:843	202, 201	202	102	5	53
8	12/07/2009 12:23:59:562	202, 201	202	102	6	52
11	12/07/2009 12:24:20:250	202, 201	202	102	7	53
14	12/07/2009 12:24:40:750	202, 201	202	102	8	54
17	12/07/2009 12:25:01:296	202, 201	202	102	9	53
20	12/07/2009 12:25:22:046	202, 201	202	102	10	52
23	12/07/2009 12:25:42:656	202, 201	202	102	11	50
26	12/07/2009 12:26:03:000	202, 201	202	102	12	52
29	12/07/2009 12:26:23:750	202, 201	202	102	13	53
32	12/07/2009 12:26:44:406	202, 201	202	102	14	52
35	12/07/2009 12:27:04:921	202, 201	202	102	15	49

Figure 4 The log for the first demonstration sorted by Originating MoteID

In all the following demonstrations the network layout was changed so that the digital forensic motes were able to see all other motes in the network. The purpose of these demonstrations is to observe the versatility of the FWSN. The second demonstration focuses on the ability of the FWSN to capture data packets from WSNs consisting of different hardware, but which are operating on the same frequency range.

Line	Date	Seen By	Forensic MoteID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
3	12/07/2009 13:02:22:484	201, 202	201	126	0	47	125	31	18	0
5	12/07/2009 13:02:42:328	201, 202	201	126	0	47	125	31	18	0
7	12/07/2009 13:03:00:500	201, 202	201	126	0	47	125	31	18	0
11	12/07/2009 13:03:29:046	201, 202	201	126	0	47	125	31	18	0
13	12/07/2009 13:03:47:046	201, 202	201	126	0	47	125	31	18	0
16	12/07/2009 13:04:11:453	201, 202	201	126	0	47	125	31	18	0
18	12/07/2009 13:04:29:984	201, 202	201	126	0	47	125	31	18	0
22	12/07/2009 13:06:09:437	201, 202	201	126	0	47	125	31	18	0
23	12/07/2009 13:06:37:000	201, 202	201	126	0	47	125	31	18	0
1	12/07/2009 13:02:07:718	201, 202	201	126	0	102	255	32	1	0
2	12/07/2009 13:02:21:390	201, 202	201	126	0	102	255	32	2	0
4	12/07/2009 13:02:34:843	201, 202	201	126	0	102	255	32	3	0
6	12/07/2009 13:02:48:453	201, 202	201	126	0	102	255	32	4	0
8	12/07/2009 13:03:02:000	201, 202	201	126	0	102	255	32	5	0
9	12/07/2009 13:03:15:359	201, 202	201	126	0	102	255	32	6	0
10	12/07/2009 13:03:28:796	201, 202	201	126	0	102	255	32	7	0
12	12/07/2009 13:03:42:406	201, 202	201	126	0	102	255	32	8	0

Figure 5 Demonstration to show the FWSN captures packets from separate OWSNs simultaneously.

It is clear from figure 5 that all the labels at the top are now labeled “Byte 1” to “Byte 7”. The columns actually go up to “Byte 64”, which is the maximum size of any packet that can be sent over a WSN using the Crossbow Imote2 hardware. In the figure, Byte 3 is the originating MoteID and Byte 5 is the packet length. For this demonstration a Crossbow TelosB mote was used as an OWSN mote. This TelosB mote belongs to students in the engineering department and the authors have no knowledge as to what the actual purpose was of the mote, as we were only interested in if we can capture the data. This mote had been assigned a MoteID of 47, which was assigned by someone else, as no alterations were made on the mote for the demonstrations. One of the motes, with MoteID 102, from the previous demonstration was also used as an OWSN mote. The OWSN, thus, consisted of these two motes. This

demonstration was performed to prove that the FWSN is able to capture data from any type of WSN with the single requirement that the communication between the motes is within the radio frequency range that the digital forensic readiness network is capable of monitoring. One can see from figure 5 that the FWSN motes were successfully able to capture the data from both the MoteID 47 and MoteID 102 as indicated by the “Byte 3” column. In the case where the OWSN radio frequency falls outside the scope of the FWSN radio frequency range, the software can simply be installed on other types of FWSN motes which would be capable of the required frequency range.

In the third and final demonstration, which can be observed in figure 6, it is shown that the FWSN can be used to confirm the integrity of the data as captured from the OWSN. This demonstration was carried out by taking a single OWSN mote and initially placing it in a location where it received direct sunlight so that its temperature sensor can measure the heat at this location. After a while, this OWSN sensor was moved into a cool, dark location. The fluctuations of the temperature were then seen on the OWSN logs, but the FWSN logs provided a backup of this data to confirm the authenticity and integrity of the data. These fluctuations can be seen in figure 6 where it is indicated by the last column labeled “Temperature Value”,

Line	Date	Seen By	Forensic MoteID	Byte 1	Byte 2	Originating MoteID	Byte 4	Byte 5	Byte 6	Byte 7	Light Value	Temp Value
4	12/07/2009 12:52:08:734	202, 201	202	126	0	101	255	32	4	0	24	23
5	12/07/2009 12:52:18:912	202, 201	202	126	0	101	255	32	5	0	24	24
6	12/07/2009 12:52:29:234	202, 201	202	126	0	101	255	32	6	0	24	25
7	12/07/2009 12:52:39:625	202, 201	202	126	0	101	255	32	7	0	24	26
8	12/07/2009 12:52:49:912	202, 201	202	126	0	101	255	32	8	0	24	27
9	12/07/2009 12:52:59:862	202, 201	202	126	0	101	255	32	9	0	24	29
10	12/07/2009 12:53:10:515	202, 201	202	126	0	101	255	32	10	0	24	30
11	12/07/2009 12:53:20:828	202, 201	202	126	0	101	255	32	11	0	24	31
12	12/07/2009 12:53:30:890	202, 201	202	126	0	101	255	32	12	0	24	31
13	12/07/2009 12:53:41:328	202, 201	202	126	0	101	255	32	13	0	24	32
14	12/07/2009 12:53:51:703	202, 201	202	126	0	101	255	32	14	0	24	32
15	12/07/2009 12:54:01:921	202, 201	202	126	0	101	255	32	15	0	24	33
16	12/07/2009 12:54:12:187	202, 201	202	126	0	101	255	32	16	0	24	33
17	12/07/2009 12:54:22:406	202, 201	202	126	0	101	255	32	17	0	24	34
18	12/07/2009 12:54:32:859	202, 201	202	126	0	101	255	32	18	0	24	35
19	12/07/2009 12:54:43:000	202, 201	202	126	0	101	255	32	19	0	24	35
20	12/07/2009 12:54:53:328	202, 201	202	126	0	101	255	32	20	0	24	36
21	12/07/2009 12:55:03:887	202, 201	202	126	0	101	255	32	21	0	24	36
22	12/07/2009 12:55:14:156	202, 201	202	126	0	101	255	32	22	0	0	33
23	12/07/2009 12:55:24:265	202, 201	202	126	0	101	255	32	23	0	0	32
24	12/07/2009 12:55:34:546	202, 201	202	126	0	101	255	32	24	0	1	30
25	12/07/2009 12:55:44:921	202, 201	202	126	0	101	255	32	25	0	0	29
26	12/07/2009 12:55:55:328	202, 201	202	126	0	101	255	32	26	0	0	28
27	12/07/2009 12:56:05:453	202, 201	202	126	0	101	255	32	27	0	0	27
28	12/07/2009 12:56:15:796	202, 201	202	126	0	101	255	32	28	0	1	26
29	12/07/2009 12:56:26:140	202, 201	202	126	0	101	255	32	29	0	0	25
30	12/07/2009 12:56:36:484	202, 201	202	126	0	101	255	32	30	0	0	24
31	12/07/2009 12:56:46:656	202, 201	202	126	0	101	255	32	31	0	0	24
32	12/07/2009 12:56:57:140	202, 201	202	126	0	101	255	32	32	0	1	23
33	12/07/2009 12:57:07:265	202, 201	202	126	0	101	255	32	33	0	0	22
34	12/07/2009 12:57:17:703	202, 201	202	126	0	101	255	32	34	0	0	22

Figure 6 Demonstration to show the FWSN captures packets with authenticity and integrity intact

This shows that adding digital forensic readiness to an existing WSN added the benefit that one could monitor the OWSN to confirm the data which was received by the motes of the OWSN or for troubleshooting faults in the OWSN.

The next section concludes the paper by discussing the strength of the implementation and how it has provided digital forensic readiness to an existing network.

V. CONCLUSION

The wireless sensor network is a type of network which makes it very difficult for any type of digital forensic analysis

due to the nature of the network. This paper proposed a digital forensic readiness prototype implementation which can be deployed on top of an existing IEEE 802.15.4 WSN to provide a layer of digital forensic readiness to the network without modifying the original network.

This paper focused mainly on being able to add a digital forensic readiness network to any existing WSN without having previous knowledge of the existing network or having to modify it. The problem that this paper addressed was that currently there exists only a list of requirements in order to achieve digital forensic readiness in a wireless sensor network, and these requirements have not yet been tested by means of a prototype and with real world wireless sensor network scenarios. This paper has shown that using the list of requirements in order to design a prototype is beneficial whilst adding digital forensic readiness to any WSN environment. This paper showed that the prototype was able to successfully provide a digital forensic layer to any existing WSN whilst adhering to all the digital forensic readiness requirements that have been set out.

As this is only a prototype implementation it has only been demonstrated by the use of four nodes. In future research this can be expanded to a larger network with more nodes. This can be achieved by using network simulator tools such as NS2 and OMNET++. The authors will also explore the ability to detect flooding attacks using the digital forensic WSN. The digital forensic WSN should be able to easily determine if there is a sudden influx of data packets on the network at a certain point in the network. This will be examined by making use of demonstrations as were used in this paper and further modifications to the protocol. We, as the authors, feel that the flooding attack on WSNs still needs a substantial amount of attention.

REFERENCES

- [1] C Chong and S P Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247-1256, 2003.
- [2] F Mouton and H S Venter, "A Secure Communication Protocol for Wireless Sensor Networks," in *Proceedings of the Annual Security Conference "Security Assurance and Privacy: organizational challenges"*, Las Vegas, 2009.
- [3] D Estrin, L Girod, G Pottie, and M Srivastava, "Instrumenting the world with wireless sensor networks," in *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2001, pp. 2033-2036.
- [4] W Ye, J Heidemann, and D Estrin, "An energy-efficient MAC Protocol for wireless sensor networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communication Societies*, 2002, pp. 1567-1576.
- [5] W R Heinzelman, J Kulik, and H Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, 1999, pp. 174-185.
- [6] K Sohrabi, J Gao, V Ailawadhi, and G J Pottie, "Protocols for self-organization of a wireless sensor network," *Personal Communications, IEEE Wireless communications*, vol. 7, no. 5, pp. 16-27, October 2000.
- [7] J Elson and D Estrin, "Time synchronization for wireless sensor networks," in *Proceedings of the 15th International Symposium on Parallel and Distributed Processing*, 2001, pp. 1965-1970.
- [8] J M Kahn, R H Katz, and K S Pister, "Next century challenges: mobile networking for "Smart Dust"," in *In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, New York, 1999, pp. 271-278.
- [9] F Zhao, J Shin, and J Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Magazine*, vol. 19, pp. 61-72, 2002.
- [10] A Mainwaring, D Culler, J Polastre, R Szewczyk, and J Anderson, "Wireless sensor networks for habitat monitoring," in *In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, New York, 2002, pp. 88-97.
- [11] C Lu, B M Blum, T F Abdelzaher, J A Stankovic, and T He, "RAP: a real-time communication architecture for large-scale wireless sensor networks," in *Proceedings of the 8th IEEE International Workshop on Real-Time and Embedded Technology and Applications Symposium*, 2002, pp. 55-66.
- [12] J Tan, Forensic Readiness, 2001.
- [13] R Rowlingson, "A Ten Step Process for Forensic Readiness," *International Journal of Digital Evidence*, vol. 2, no. 3, 2004.
- [14] C P Grobler, C P Louwrens, and S H von Solms, "A Framework to Guide the Implementation of Proactive Digital Forensics in Organisations," in *International Conference on Availability, Reliability and Security*, Los Alamitos, CA, USA, 2010, pp. 677-682.
- [15] F Mouton and H S Venter, "Requirements for Wireless Sensor Networks in order to achieve Digital Forensic Readiness," in *6th International Annual Workshop on Digital Forensics & Incident Analysis*, London, 2011.
- [16] Crossbow Technology Inc, *Imote2 Hardware Reference Manual*, Revision A ed. San Jose: Crossbow Technology Inc, 2007.