A Secure Patient Monitoring Solution using Wireless Sensor Networks

Submitted by Avijit Mathur
For the award of Doctor of Philosophy

Supervised by Dr. Thomas Newe
Department of Electronic and Computer Engineering

Submitted to the University of Limerick
October 2017
Declaration

This thesis is written to meet the requirements for the degree of Doctorate of Philosophy. It is entirely my own work and has not been submitted to any other university or higher institution. Where the work of other people has been used, it has been fully referenced and acknowledged.

Signed ____________________________________

Avijit Mathur
# Contents

1 INTRODUCTION .......................................................... 1

1.1 RESEARCH OBJECTIVES .............................................. 3

1.2 RESEARCH METHODOLOGY .......................................... 4

1.3 NOVELTY OF RESEARCH ........................................... 5

1.4 STRUCTURE OF THESIS ............................................... 5

1.5 RESULTING PUBLICATIONS ........................................... 6

1.6 SUMMARY ................................................................... 6

2 SENSOR NETWORKS BACKGROUND AND OVERVIEW ........... 9

2.1 WIRELESS SENSOR NETWORKS ..................................... 9

2.1.1 Existing healthcare Systems ...................................... 10

2.1.2 Applications .......................................................... 12

2.1.3 Protocols .............................................................. 14

2.1.4 Platforms ............................................................. 24

2.1.5 Operating Systems .................................................. 27

2.2 SUMMARY .................................................................. 33

3 SAS: SYSTEM AND SECURITY ........................................ 35

3.1 SYSTEM OVERVIEW .................................................. 35

3.1.1 Security Protocols .................................................. 37

3.1.2 Communication Protocols ......................................... 39

3.2 SECURITY OVERVIEW ................................................ 42

3.2.1 Security Services ................................................... 42

3.2.2 Security Attacks ..................................................... 43

3.2.3 Physical Security .................................................... 44

3.2.4 Link Layer Security ............................................... 45

3.2.5 Routing Security .................................................... 46

3.2.6 Internet Security .................................................... 48

3.3 SUMMARY .................................................................. 50

4 SECURE ROUTING ........................................................ 51

4.1 CURRENT SOLUTIONS ................................................ 52

4.2 EXPERIMENTAL SET-UP ............................................. 54

4.2.1 Routing .................................................................. 55

4.2.2 Black-hole attack and consequences ............................ 56

4.2.3 Black-hole Countermeasure ...................................... 57

4.2.4 Selective Forwarding attack and consequences ............... 59

4.2.5 Selective Forwarding Countermeasure ......................... 59

4.2.6 Packet Gathering .................................................... 63

4.2.7 BS Analysis .......................................................... 64
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>LINUX-PC APPLICATION (MAIN FUNCTION)</td>
<td>121</td>
</tr>
<tr>
<td>B.2</td>
<td>EDISON BOARD (SAMPLE SERIAL DATA)</td>
<td>122</td>
</tr>
<tr>
<td>B.3</td>
<td>CLUSTERSCHEM.H (SENSOR NODES)</td>
<td>123</td>
</tr>
<tr>
<td>B.4</td>
<td>KEYS.H (ROUTER NODES)</td>
<td>125</td>
</tr>
</tbody>
</table>

**BIBLIOGRAPHY** | 127
List of Figures

FIGURE 2.1: OVERVIEW OF A WIRELESS SENSOR NETWORK SYSTEM IN HEALTHCARE .............................................10
FIGURE 2.2: SENSORS USED FOR MEDICAL APPLICATION OF WIRELESS SENSOR NETWORKS ...........................................12
FIGURE 2.3: WIRELESS SENSOR NETWORKS FOR STRUCTURAL HEALTH MONITORING .............................................13
FIGURE 2.4: MILITARY APPLICATION FOR WIRELESS SENSOR NODES ........................................................................14
FIGURE 2.5: IEEE 802.15.4 STACK ..................................................................................................................16
FIGURE 2.6: IEEE 802.15.6 STACK ..................................................................................................................17
FIGURE 2.7: IEEE 802.15.4G STACK ..................................................................................................................17
FIGURE 2.8: BLUETOOTH LOW ENERGY STACK ............................................................................................18
FIGURE 2.9: LORA PROTOCOL STACK ..............................................................................................................19
FIGURE 2.10: SIGFOX PROTOCOL STACK ............................................................................................................20
FIGURE 2.11: Z-WAVE PROTOCOL STACK .........................................................................................................21
FIGURE 2.12: ZigBee COMMUNICATIONS PROTOCOL STACK COUPLED WITH IEEE 802.15.4 STACK ...............22
FIGURE 2.13: TMOTE SKY ......................................................................................................................................25
FIGURE 2.14: WASPMOTE, (A) TOP VIEW, AND (B) FRONT DIAGONAL VIEW ....................................................26
FIGURE 2.15: SHIMMER3 NODES ..........................................................................................................................26
FIGURE 2.16: OPENMOTE, (A) OPENMOTE XBEE BREAKOUT CHIP, AND (B) OPENBASE BOARD ....................27
FIGURE 2.17: SPLIT-PHASE REQUEST FOR SERVICE IN TINYOS ........................................................................30
FIGURE 2.18: CONTIKI OS SYSTEM PARTITIONING ............................................................................................32
FIGURE 3.1: END-TO-END SECURE WSN SYSTEM – I ............................................................................................35
FIGURE 3.2: COMMUNICATION AND SECURITY PROTOCOLS IMPLEMENTED ..................................................37
FIGURE 3.3: STEPS FOR THE ECDH ALGORITHM ..............................................................................................38
FIGURE 3.4: AES GCM MODE OF OPERATION ..................................................................................................39
FIGURE 3.5: RIME COMMUNICATION STACK ...................................................................................................40
FIGURE 3.6: MQTT PUBLISH/SUBSCRIBE MODEL ..............................................................................................41
FIGURE 3.7: UART ON A COMPUTER/GATEWAY CONNECTED TO A SERIAL DEVICE (SAY OPENMOTE) ............42
FIGURE 3.8: TLS CLiENT/SERVER HANDSHAKE BETWEEN INTEL EDISON, AND MOSQUITTO BROKER ....49
FIGURE 4.1: SYSTEM MODEL .............................................................................................................................51
FIGURE 4.2: NETWORK MODEL ..........................................................................................................................54
FIGURE 4.3: ROUTING PHASE: (A) AODV PROTOCOL, (B) DATA DIRECTION, AND (C) MODIFICATION FOR OUR SYSTEM ..............................................................................................................................55
FIGURE 4.4: BLACK HOLE ATTACK SIMULATION ON COOJA SIMULATOR (CONTIKI): (A) NODES LAYOUT.
SOURCE NODE GREEN ‘2’, DESTINATION IS BLUE ‘1’ AND MALICIOUS NODE IS RED ‘5’, (B) MOTE
OUTPUT: NODE ‘2’ SENDING DATA PACKETS TO NODE ‘1’ VIA NODE ‘5’ ................................................................56
FIGURE 4.5: PRE-DEPLOYMENT PROCESS ............................................................................................................57
FIGURE 4.6: RREQ PACKET ....................................................................................................................................58
FIGURE 4.7: PHASES OF SELECTIVE FORWARDING (SF) COUNTERMEASURE. IN ORDER FROM LEFT TO
RIGHT. ...........................................................................................................................................................................60
FIGURE 4.8: NEIGHBOUR MONITORING PROCESS ...............................................................................................60
FIGURE 4.9: PACKETS: (A) DATA PACKET, (B) ACK PACKET ..................................................................................61
List of Tables

TABLE 2.1: STANDARD COMMUNICATION PROTOCOLS ................................................................. 23
TABLE 2.2: NEW COMMUNICATION PROTOCOLS ........................................................................ 24
TABLE 2.3: OPERATING SYSTEMS SPECIFICATIONS ................................................................. 33
TABLE 3.1: SECURITY ATTACKS BASED ON LAYERS OF THE OSI MODEL ................................. 44
TABLE 4.1: FLAGS BIT ARRAY – PASS I ......................................................................................... 65
TABLE 4.2: FLAGS BIT ARRAY – PASS II ......................................................................................... 65
TABLE 4.3: SINGLE SELECTIVE FORWARDING ........................................................................... 68
TABLE 4.4: COLLABORATIVE SELECTIVE FORWARDING ............................................................. 68
TABLE 4.5: ACCURACCY CHECK FOR SINGLE SF - NULLRDC .................................................. 69
TABLE 4.6: ACCURACY CHECK FOR SINGLE SF - CONTIKIMAC RDC ......................................... 69
TABLE 4.7: ACCURACY CHECK FOR COLLABORATIVE SF - NULLRDC ....................................... 70
TABLE 5.1: RELATED WORK COMPARISON .................................................................................. 79
TABLE 5.2: ECDH PHASE I BETWEEN ROUTER R AND CLUSTER OF SENSORS S ...................... 85
TABLE 5.3: KEY MANAGEMENT TECHNIQUES ANALYSIS ........................................................... 87
TABLE 5.4: AES ANALYSIS ............................................................................................................. 88
TABLE 5.5: PRNGS SCENARIOS .................................................................................................... 89
TABLE 5.6: NIST STATISTICAL TEST SUITE RESULTS (P-VALUE) ............................................... 90
TABLE 6.1: CH ROTATION FREQUENCY ...................................................................................... 106
TABLE 7.1: SECURE ROUTING NORMAL PHASE VS DETECTION PHASE ...................................... 113
TABLE 7.2: CLUSTER ELECTIONS SIMULATOR VS HARDWARE ..................................................... 114
TABLE 7.3: CLUSTER ELECTIONS POWERTRACE VS AGILENT ON HARDWARE ......................... 115
Abbreviations

6LoWPAN  Internet Protocol version 6 over Low-power Wireless Personal Area Networks
ACK  Acknowledgement
ADV  Advertising
AES  Advanced Encryption Standard
AES-CBC  AES Cipher Block Chaining
AES-GCM  AES Galois Counter Mode
AODV  Ad-hoc On-demand Distance Vector
AP  Access Point
BLE  Bluetooth Low Energy
BS  Basestation
CA  Certificate Authority
CCA  Clear Channel Assessment
CH  Cluster Head
CIA  Confidentiality, Integrity and Availability
CNIDL  Cluster Node Identification List
CPU  Central Processing Unit
CSMA/CA  Carrier-Sense Multiple Access with Collision Avoidance
CSS  Chirp Spread Spectrum
DC  Direct Current
DLL  Data Link Layer
DSSS  Direct Sequence Spread Spectrum
ECC  Elliptic Curve Cryptography
ECDH  Elliptic Curve Diffie-Hellman
ECG  Electrocardiogram
EEG  Electroencephalogram
FTDI  Future Technology Devices International
GCM  Galois/Counter Mode
HAL  Hardware Abstraction Layer
HBC  Human Body Communications
HMAC  Hash-based Message Authentication Code
HKDF  HMAC-based Key Derivation Function
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IKM</td>
<td>Input Keying Material</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>ISM</td>
<td>Industrial, Scientific, and Medical</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>LPM</td>
<td>Low Power Mode</td>
</tr>
<tr>
<td>LR-WPAN</td>
<td>Low Rate Wireless Personal Area Networks</td>
</tr>
<tr>
<td>LTE</td>
<td>Long-Term Evolution</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MMU</td>
<td>Memory Management Unit</td>
</tr>
<tr>
<td>MQTT</td>
<td>Message Queuing Telemetry Transport</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>OKM</td>
<td>Output Keying Material</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>OQPSK</td>
<td>Offset Quadrature Phase Shift Keying</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical</td>
</tr>
<tr>
<td>PLR</td>
<td>Packet Loss Ratio</td>
</tr>
<tr>
<td>PRK</td>
<td>Pseudo Random Key</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RDC</td>
<td>Radio Duty Cycling</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comments</td>
</tr>
<tr>
<td>ROM</td>
<td>Read-Only Memory</td>
</tr>
<tr>
<td>RPL</td>
<td>IPv6 Routing Protocol for Low-Power and Lossy Networks</td>
</tr>
<tr>
<td>RREP</td>
<td>Route Reply</td>
</tr>
<tr>
<td>REQ</td>
<td>Route Request</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received Signal Strength Indicator</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real Time Operating System</td>
</tr>
<tr>
<td>RX</td>
<td>Receive</td>
</tr>
<tr>
<td>SHA</td>
<td>Secure Hash Algorithm</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>SUN</td>
<td>Smart Utility Network</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TTL</td>
<td>Time to Live</td>
</tr>
<tr>
<td>TTP</td>
<td>Trusted Third Party</td>
</tr>
<tr>
<td>TX</td>
<td>Transmit</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver/Transmitter</td>
</tr>
<tr>
<td>UDGM</td>
<td>Unit Disk Graph Medium</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Networks</td>
</tr>
</tbody>
</table>
To my
Parents, Sister, & Grandparents
Acknowledgements

This PhD has been a long journey of four years, during this time I faced many challenges in my academic and personal life. Tackling these challenges in a healthy manner has led to the completion of my thesis. Thanks to the support of many people.

I would like to express my gratitude to my supervisor Dr. Thomas Newe for guiding me in the right direction with his advice and patience.

I am extremely thankful to the Irish Research Council for providing the funding that helped me complete my research and go to conferences around Europe while staying comfortably in Ireland. Also, a big thank you to the University of Limerick for assisting with the fees of my PhD course.

I would like to thank my parents, and sister for everything because without them I would have never reached this stage in my life. And I would like to thank my favourite music band Breaking Benjamin for being there through the thick and thin of my PhD. Additionally, I would like to thank my friends and housemates: Petra, Eugénie, Jose, Sarah, Shishir, Marystela, Faris, Tony, and Eamon for being there through my journey.

Lastly, I would like to thank my colleagues Muzaffar and Walid for providing a joyful work environment through the course of my PhD.
Abstract

The Wireless sensor networks (WSN) is developing over time in terms of technologies, protocols and security. Each of these have different requirements, challenges, and solutions, which may or may not be similar for the different applications of this field like medical, civil/structural, marine, agriculture, and military.

Regarding the medical application, a complete study of the security measures required for WSN systems is discussed with a flavour of technologies and protocols used in the system. The inclusion of security is an important phase for wireless networks especially in medical domain due to the involvement of patient privacy.

The study works towards the creation of an end-to-end secure WSN solution that collects patient data via sensor nodes at one end and transports it to the other end i.e. a hospital/doctor’s PC. This communication encompasses tough security measures. These measures are applied to each component/process of the presented system: patient data, sensor nodes, routing phase, and internet connection. The security measures applied to these phases collectively achieve confidentiality, integrity, availability, authentication, DoS resistance, and strong key management features. Each of these are studied extensively, and are implemented on simulator and/or hardware platforms. The implementation presented in this work gives rise to the following results: 96.5% detection and 83% correction rate for secure routing; 21.49mJ and 0.049mJ energy consumption for Elliptic Curve Diffie-Hellman (ECDH) and Hash message authentication code based Key Derivation Function (HKDF) Key generation; and $n$ unique Cluster Heads for $n$ number of sensor nodes in the cluster phase of the presented system. These results are an improvement compared to the respective related work provided for each phase.

Finally, the study and its results increase confidence in using the presented system in hospitals and clinics for the benefit of the patients and the hospital staff.
Chapter 1

Introduction

The basis of the research presented in this thesis comes from the importance of automating certain tasks for medical monitoring systems using wireless sensor networks (WSNs). This extends to the requirement of providing optimal solutions for integrating the sensor network system into the medical field by giving solutions that deliver patient data from one end of the system to the other while keeping the data, the devices, and the network secure. This integration of WSN provides real-time patient vitals’ information to doctor/hospital staff, which can mitigate losses in life, and reduce costs [1]. This integration of automation can also reduce the amount of time a doctor may need for diagnosis [2].

Since this WSN system must transport patient data it is important to secure this system, because according to the EU law for patient data [3]:

“Data has to be processed in a manner that minimises risks to confidentiality and integrity of the data (which means ensuring its consistency and accuracy, as opposed to data corruption)”

And according to Irish Data protection act it is required to keep patient data “safe and secure”. Failure to comply shall result in a penalty of €3,000 to €100,000 for offenders [4]. Thus showing that security is very important when dealing with patients and/or their data especially when it comes to systems like WSN. The reason being WSN are wireless in nature and hence more prone to security attacks than wired networks.

During the specification of the research presented in this thesis, several systems were available for achieving tasks of patient-data delivery with limited or basic security functionality. Some of the these are: MEDIsN [5], which provides CC2420 in-line security that incorporates data confidentiality only, System in [6], which provides authentication via X.509 certificates with data confidentiality, and System in [7]–[9], which do not mention any security measures being applied to their respective systems. Each of these are explained in detail in Chapter 2, but this brief introduction on them shows that the available systems contain some basic security for confidentiality and/or authentication. However, they do not mention and/or implement higher security features like DoS resistance, which is required because a DoS attack can make a system
unavailable to the end-users thereby denying availability. For example, A distributed DoS attack was carried out on the Boston Children’s hospital by the Anonymous group in 2014 [10]. This attack was carried out on the network of the hospital resulting in no internet access for a week; patients could not access appointments and results online during this time, resulting in a loss of approximately $300,000 to the hospital. Although this attack was not specifically targeted towards a WSN network, it shows the gravity of DoS attacks on wireless and wired networks. And since WSN nodes are vulnerable to security attacks, with DoS being the most popular attack on these nodes [11] therefore, it is important to fortify the network’s security.

This motivation called for the construction of specific goals that solve the problem of delivering patient data from the patient’s body to the doctor and/or hospital staff securely. These goals are: (1) At no point of time shall the patient data be compromised, (2) The network of the system must be secure and available always, (3) The system must be as automated as possible to prevent manual intrusion.

Goal (1) is very important because privacy of patients must not be affected. Goal (2) is defined to keep any intruder out of the system, and keep the system services always available to the user i.e. doctor / hospital staff and patients. Goal (3) is formulated to reduce system maintenance by not burdening the involved users.

These goals help formulate a WSN system that provides patient monitoring facilities in a secure fashion. This will assist hospitals and clinics in obtaining real-time patient data while freeing up ward beds and keeping the involved patients’ comfort in mind.

Arising from these goals are the requirements and challenges in WSN for healthcare as outlined in [12], [13]. These encompass data privacy, and system availability, reliable data transmission, node mobility support, and Energy/power management.

- Data privacy: refers to hiding of patient data. This is required because leaking of patient data could be hazardous to their life or health [14].
- System availability: refers to the availability of system services to an end-user. This is required as without this feature the system will not be able to achieve its objective i.e. patient-data delivery.
- Reliable data transmission: refers to the transmission of data from one end to the other without loss of packets. This is required as otherwise the goal of the system i.e. patient-data delivery to destination will fail.
• Node mobility support: refers to the capability of a system to adapt to the movement of nodes. This is required as in a healthcare application patients may move around and therefore so do the sensor nodes.

• Energy/Power management: refers to optimising energy or power of the nodes. This is required as the sensor nodes and routing nodes operate on battery power.

• Automation: refers to reducing human intervention for healthcare tasks that measure patient vitals. This is required to reduce burden on the hospital staff while providing emergency care to the patients.

These points give rise to important properties that are incorporated into the presented WSN system:

• Security is integrated in every phase of the network.

• Sensor nodes / routers can adapt to the changing requirements of the cluster or network.

• Sensors nodes and/or other devices are to be operated with minimum supervision from humans.

1.1 Research Objectives

The proposed research project involves the development of a WSN system that enables body worn wireless sensors to communicate easily and securely to a hospital/doctor PC regardless of the position of the patient and the PC. This type of system is intended to support deployment of sensors in a hospital environment and at patients’ homes where the sensor technology can be utilised to monitor a range of conditions that will indicate the health or status of the patient. The entire system is constructed and is bonded together by a system of secure communications utilising all the principles of Secure WSNs. The main aim arising out for this type of system is the integration of DoS resistance, Data confidentiality and authentication, and entity authentication to the wireless sensor networking system being built for patient monitoring. Here:

• The DoS resistance is to be deployed for black-hole (BH) and selective forwarding (SF) attacks on the routing side of the WSN system. These attacks are selected as they are one of the most dangerous DoS attacks for WSN systems as described in Chapter 4.

• The entity authentication is to be deployed for internet side of the WSN system to avoid any unauthorized access, as described in Chapter 5.
• Data confidentiality and authentication is to be applied to the patient data originating at the sensors’ side of the WSN system due to privacy concerns as shown in Chapter 6.

• Energy/Power is optimized for managing the system resources via usage of radio duty cycling (RDC) for routing and sensing nodes, and cluster load balancing in Chapter 6.

• Finally, all these features are being constructed while simultaneously building the presented system that incorporates on-demand routing, cluster elections, key management implementation, WSN — Internet connection, and end-user GUI application.

1.2 Research Methodology
As seen from the discussion in Section 1 and Section 1.1, Security is very important in active health monitoring systems. Therefore, the research undertaken studies and understands the current technology used with WSNs that can offer good end-to-end reliable and secure communications necessary for medical monitoring systems.

Keeping the security at focus, the research implements secure routing, key management, and cluster security features for the WSN system being constructed. This research also encompasses protocols for efficient, reliable and secure transmission in WSNs considering the resource constraints inherent to these networks.

These goals are achieved by the following processes:

• Simulating DoS attack for Secure Routing phase on contiki cooja simulator that helps in the formation of its countermeasure that runs on both simulator and hardware.

• Deploying Key management protocols for periodically updating encryption keys. This is useful for preventing permanent key compromise. The technologies used are Elliptic Curve Diffie-Hellman (ECDH) and HMAC-based Key Derivation Function (HKDF) across multiple platforms. The former is implemented with assistance from the sensor nodes’s libraries that use the nodes’ Elliptic Curve Hardware accelerator, while the latter is implemented as a contiki application by the author.

• Simulating and deploying cluster of sensors that represent sensors attached to a patient’s body. For testing purposes, these sensors capture data like temperature,
humidity and luminosity values of the room in which they reside instead of capturing actual patient data. This is done to speed up the testing processes as acquiring patient data would require ethics approval, and several patients consent. The usage of room-sensed values is justified because the data received at sensor’s side will be treated in same fashion regardless of the type of sensors being used. Additionally, the conversion of raw data to stored sensor data is out of scope for the research conducted in this thesis.

- Connecting the sensor side of the system to doctor’s PC via Internet. This part of the network is secured with username, password, and Transport Layer Security (TLS).

1.3 Novelty of Research

The novelty of this research can be categorised as follows:

- The system is secure end-to-end i.e. it provides network security, internet-security, and data security. To the authors’ knowledge, the systems at the time of development provided only authentication and/or data confidentiality, thus leaving out network and/or internet security.

- The security extends to a modified on-demand routing protocol in Chapter 4 that defends the system against BH and SF attacks, and it is capable of automatically forming a new path free of compromised nodes. This implementation makes the network self-healing and self-organising while being secure. To the author’s knowledge this type of techniques have not been used/deployed, with a high detection rate (96.5%) and correction rate (83%), in an end-to-end WSN system for medical application.

- The system incorporates key management features ECDH and HKDF that allow key updates across several platforms and communication channels. The smooth functioning of these technologies with low energy consumption and fast times (see Table 5.3) across WSN and Internet is not found in previous related systems.

- Finally, the system is implemented on both simulator and hardware thereby increasing confidence in the results. It also presents a GUI that interacts with the user to control and manage the network. This makes the system more organized and presentable for real-world usage.

1.4 Structure of Thesis

Chapter 2 provides a background on the field of WSNs. It takes the reader through a journey of sensor networks encompassing their applications, protocols and available
platforms. It then provides justification for the selection of parameters for the same in the implemented system. Furthermore, the chapter presents detailed description of sensor networks operating systems (OS), and compares these in-order to finalise the selection of the OS for the implemented system.

Chapter 3 outlines the implemented system, and associated protocols. It then reviews the security parameters and challenges that encompass the sensor networks, the medical application and the implemented system.

Chapter 4 presents a secure on-demand routing protocol that defends the system against denial of service (DoS) attacks like black-hole and selective forwarding. The chapter provides information on the attacks, and solutions for preventing and/or correcting them with analysis for each stage of the solution. Finally, it provides several results that are used to prove the importance of this protocol over other available alternate solutions.

Chapter 5 presents the key management techniques that are deployed in the implemented system. The chapter provides the reason for doing so, security analysis, and the associated results that show an improvement in the functioning of the system compared to other available solutions. Additionally, it presents a new Pseudo Random Number Generator Seeding (PRNGS) for the improvement of key management techniques.

Chapter 6 presents Cluster Head elections with cluster security. It provides available alternate solutions for this phase, and then presents the phase with focus on architecture, and the experimental set-up. Next, the chapter provides results for this phase in terms of power, energy and current consumption. Finally, it summarises the phase by comparing the results with the available solutions.

Chapter 7 concludes the thesis by highlighting the research undertaken, its comparison on simulator and hardware, its contributions to the field, and the future work.

1.5 Resulting Publications
Several technical papers have been published throughout the course of this research. Each of these compile the research done at that point of time in the PhD. These include six conference papers, two journal papers, and one book chapter. A detailed list of these publications is presented in Appendix A.

1.6 Summary
This chapter introduced the research presented in this thesis. It has presented the main objectives that were formulated for the implemented system. The chapter then discusses
the goals of the system, and why they are important with focus on the novelty of this research. Finally, the chapter presents the structure of the thesis and the publications that resulted from the work undertaken.
Chapter 2

Sensor Networks Background and Overview

Wireless sensor networks (WSNs) have been in use for different applications like Medical, military, civil, marine, home, and agriculture. Each application has different requirements like number of sensors, user services, network management, reliability and security, but the fundamental requirements for all are similar i.e. a network of sensors connected in a smart fashion collecting different types of data for automating mundane tasks. This chapter will discuss an introduction to wireless sensor networks with focus on its applications, communication protocols, platforms and operating systems. This introduction will lay a foundation understanding of WSNs, which in turn prepares the reader in understanding the system presented in chapter 3.

The rest of the chapter is structured as follows: section 2.1 introduces wireless sensor networks and its components. Section 2.1.1 presents the available systems for healthcare, Section 2.1.2 describes the medical and other applications for WSNs, Section 2.1.3 describes the standards and communication protocols that may be used in different WSN applications, Section 2.1.4 shows the common available hardware platforms, and Section 2.1.5 provides the popular operating systems that are currently being used for WSN applications. Finally, section 2.2 summarises this chapter.

2.1 Wireless sensor networks (WSN)

Wireless sensor networks are an array of autonomous sensor devices that are responsible for monitoring and collecting different types of data [15]. These devices may have different capabilities, and depending upon their capabilities WSNs can be used for different fields like healthcare, marine, civil or industrial, military, management and so on. In a healthcare system the type of data may vary depending upon the sensor i.e. pulse oximeter, ECG, or body temperature etc. However, in the presented system the sensors capture data like room temperature, humidity and luminosity (lux) values. This is because data acquiring and data conversion, which is done on the sensor node side, is out of scope for this research. Therefore, the presented WSN system makes use of this general data of temperature, humidity and lux values.

Additionally in conjunction with the presented WSN system, sometimes a broader term named Internet of things (IoT) may be used for reference in related work of certain parts
of thesis. Here, IoT refers to sensors or camera or any electronic device directly connected to the internet without any routing features i.e. in one-hop. For example, in the presented system, similar to one in figure 2.1, the Sink/Gateway can be called an IoT device because it sends data to a PC via Internet in one hop but the sensor nodes are WSN. However, since IoT is too broad and generalised compared to WSN the presented system and all of its components are described as WSN only.

Figure 2.1 presents a WSN network for healthcare, consisting of wireless sensor nodes, router nodes, Sink/Gateway, and Internet. Following is the description and working of each component:

1) Sensor Nodes: acquire sensor data like ECG, EEG, blood saturation level or body temperature etc. They are also responsible for confidentiality and/or authentication of data using the security mechanisms discussed in Chapter 6.

2) Routers: are responsible for routing information between two end points. This part of the network is open to routing attacks that are prevented in the implemented system using the mechanisms discussed in Chapter 4. The network topologies that may be used for routing are mesh, line, tree, and star among others.

3) Sink/Gateway: manages the network through route formation, maintenance, and collects data from the sensors. In addition, the BS may communicate with a gateway for bridging with the Internet as shown in Chapter 3.

4) Internet: here refers to a connection to another network to which the WSN might be connected. In our case, the patient monitoring system will be connected to the internet as discussed in Chapter 3 and Chapter 5.

2.1.1 Existing healthcare Systems

Some of the available WSN systems for medical applications are:
1) MEDiSN [5]: is a system capable of monitoring the physiological data of the patients in an emergency room via Physiological Monitors and Relay Points. The physiological monitors are responsible for data confidentiality and source authentication while collecting data. The relay points are responsible for the underlying routing that is in the form of a self-organising routing tree protocol, and allows for reliable delivery of periodic data and alerts. As seen, The MEDiSN system enforces security schemes via encryption of physiological data, and authentication of clients, but does not address network security.

2) UbiMon [8]: is a system capable of continuously monitoring patients in their natural states. The system notifies appropriate users of any adverse events that may take place. It consists of two types of sensors i.e. wearable and implantable. These are connected via an ad-hoc network. The system, however, does not address security schemes.

3) System in [6]: is a hybrid system architecture combining grid architecture with WSN for providing health monitoring of a heterogeneous group of patients like post-operative patients, elderly patients, patients affected by Parkinson’s disease and chronic obstructive pulmonary disease. The results are then used for the benefit of prognosis, diagnosis, and drug delivery. The system provides resource management, data management, information services, and grid security infrastructure. The security infrastructure incorporates secured communications and user authentication via X.509 proxy certificates.

4) System in [7]: is a system with multi-hop patient data delivery feature that is achieved via relay nodes and base station. The main features of this system are:
   a. The relay nodes are not required to be connected to a wired network,
   b. Patient mobility is considered and solutions are integrated into the architecture, and
   c. Radio power management on the nodes is available and simple.

However, there is no section in the paper that addresses any form of system security except remote login for diagnostics of the network or data.

5) System in [9]: consists of a network of relay nodes, and base station capable of sending data from one end to the other via multi-hop mechanism. The system is capable of monitoring wide range of patient parameters like heart-beat, blood pressure etc. for a long period of time. It also has a feature where an SMS/E-mail may be send to the concerned parties if a patient experiences adverse health
condition(s). The system consumes low energy and has extended communication coverage. However, it does not address any security measures for patient data or involved users or the network itself.

2.1.2 Applications

Wireless sensor networks have a wide range of applications. This is because they provide flexibility, self-healing and self-organising capabilities among others.

The popular applications are:

1) **Medical**: A Medical based wireless sensor network [16] is capable of performing activities that help reduce the workload for duty staff and provide more freedom for patients. Research into the use of WSNs for healthcare applications is ongoing in the area of technologies, methods and mechanisms. For example, normal monitoring of vitals, capsule endoscopy, monitoring of body motion for Parkinson’s diseases, early warning systems predicting attacks like EEG for reporting symptoms of Epilepsy, ECG for measuring heart functions and pulse oximeter for blood-saturation levels. Some of the related sensor nodes' locations are shown in Figure 2.2.

![Diagram of medical sensor nodes](image)

**Figure 2.2**: Sensors used for medical application of Wireless Sensor Networks

2) **Marine**: A Marine-based WSN may be used for monitoring purposes. This is useful because Marine-based systems are susceptible to human activities, and the network can assist in improving the real-time data associated with large geographical areas [17]. There has been considerable research in this field for example the authors in [18] implement a security scheme tailored specifically for the marine based WSNs, and the authors in [19] implement a smart sensor network designed for the marine environment that evaluates the quality of water, and captures worst-case scenarios or events while collecting long-term data.
3) **Civil, and Industrial**: WSNs may be used to monitor man-made structures, Figure 2.3. They can be used to verify the health of the buildings, its support-frames, and the rest of the infrastructure. The goal is to avoid maintainable hazards, and improve safety and reliability of the structures [20]. In this application, the WSNs are much more beneficial compared to wired monitoring systems because of lower costs, less maintenance requirements, and less weight due to absence of wires. However, these networks need to be more secure when compared to marine environment, this is because the sensor network may be managing/monitoring critical points of the structures [21].

![Figure 2.3: Wireless sensor networks for structural health monitoring](image)

4) **Military**: WSNs may also be used for military applications where the purpose ranges from surveillance of the battlefield to monitoring enemy forces [21]. They can also be used for communication operations, which refers to command distribution, intelligence or just sensor data [22]. In addition, these networks require higher security than any other application due to the nature of their work. The security features must include jamming resistance, and denial of service prevention in conjunction with solid confidentiality and authentication. Furthermore, the sensor network must provide high reliability to prevent any mishaps during sensitive operations.

A common militarized operation being monitored by sensor nodes, and cameras is shown in figure 2.4. Here, different types of sensors work together to allow the troops to have an advantage in the battlefield.
5) **Home**: Home applications of WSNs include smart homes, where sensors are added to monitor or control indoor heating, lighting, and various other in-house systems. The sensors can also be used in conjunction with medical monitoring system to provide at-home patient monitoring.

![Military Application for Wireless sensor nodes. Here the nodes transmit data to the bunker, which forwards it to the control centre via satellite communications](image)

6) **Agriculture**: applications can have a wide variety of range from soil health monitoring to meat tracking (farm to table). Different sensor systems can provide information to farmers about the health of their animals and crops, and to the end-user about the origin of their food, date of production and a feedback system. This system can provide farmers and agricultural companies with valuable input on improving the quality of their products.

### 2.1.3 Protocols

The next step for the foundation of WSNs is looking at its associated protocols. There are several existing communication protocols and standards relevant to the WSN field: 802.15.4, 802.15.6, and 802.15.4g standards, and Bluetooth Low Energy, SigFox, LoRa, Z-Wave, THREAD, TV White spaces, and ZigBee communication protocols. The features outlined suit the requirements of the implemented system in the thesis.
IEEE 802.15.4
This standard defines Physical (PHY) and Medium Access (MAC) layers for Low Rate Wireless Personal Area Networks (LR-WPANs) as illustrated in figure. 2.5. It forms the foundation for ZigBee, WirelessHART, and THREAD [23] specifications among others. It can also be used in conjunction with Internet Protocol version 6 (IPv6) over Low-power Wireless Personal Area Networks (6LoWPAN) [24]. The standard supports low power consumption, dynamic device addressing and low latency device support. The low power consumption is achieved by a reduction in duty cycle with the help of super-frame structures. These structures allow the sensor nodes to sleep during their period of inactivity [25].

The standard supports three alternative PHY radio frequency schemes: 2.4 GHz, 915 MHz (US) and 868 MHz (EU). The 2.4 GHz scheme is the most common and standard Industrial, Scientific, and Medical (ISM) band. It is used with many standards, protocols, and appliances, therefore, it is susceptible to interference. According to an experiment conducted by [26] a microwave oven causes a relatively higher interference, compared to other appliances, with an approximate packet error rate of 4% compared to 3% in Wi-Fi (802.11) and 2% with Bluetooth or absent interfering source. This interference is increased when the distance between the source and receiver is small and vice-versa for the same distance between transmitter and receiver. Finally, the channel selection and angle between source-receiver affects the system as well. This implies that the 2.4 GHz scheme might not be apt for communications in environments involving frequent use of this band for other purposes. However, since this frequency scheme is applicable for shorter distances and is compatible with most radios, therefore it is an apt choice for at-home or hospital based wireless sensor network.

Considering the 868/915 MHz bands, the former allows only one communication channel for ZigBee units and is restricted to the European area, while the latter allows up to thirty channels but is geographically restricted to North America only [27]. They both support the Direct Sequence Spread Spectrum (DSSS) modulation technique with binary or offset quadrature phase shift keying (BQPSK or OQPSK).
IEEE 802.15.6

The authors in [28] define a new standard for the specification of wearable WSNs. It incorporates a compulsory safety procedure and a new PHY layer i.e., Human Body Communications (HBC), Figure 2.6. The standard supports three operational PHYs. The main responsibilities of these layers are 1) radio transceiver activation/deactivation, 2) Clear Channel Assessment (CCA) and 3) data transmission and receiving. Moreover, it defines a new MAC layer in which resource allocation is contended using CSMA/CA or slotted Aloha access procedure. This standard is a step forward in wearable wireless sensor networks as it is designed specifically for use with a wide range of data rates, less energy consumption, low range, ample number of nodes (256) per body area network and different node priorities according to the application requirements.

The standard supports three security schemes 1) unsecured communications level wherein there is no form of security i.e. authentication or encryption. 2) Authentication only can prove the authenticity of the data source and 3) authentication and encryption provides confidentiality of data in addition to its authentication [29]. Hence, it provides flexibility in security features as encryption might not be required in certain cases when two parties exchange public information (say public keys).

In [30] the performance of 802.15.6 in comparison to 802.15.4 is evaluated. This reveals that the former has a lower packet loss rate (PLR) when the payloads are long and vice-versa when payloads are short. However, 802.15.6 incurs more delays compared to 802.15.4, but with added security features and options. Therefore, it is inferred that 802.15.6 may be apt for a hospital environment, where more data is to be transmitted, while 802.15.4 is better for personal use by individual patients at home.
IEEE 802.15.4g

This is a brand new standard [31], designed for smart utility networks (SUN) by defining a robust physical (PHY) layer for outdoor wireless communications as shown in figure 2.7. This design is directed towards low cost, and low power devices with data rates ranging from 40kbps to 1Mbps, and a frame size of up to 1500 bytes. The target use cases for SUN are gas metering, demand/response, and distribution automation.

The standard has certain similarities with the IEEE 802.11 (Wi-Fi):

- Bit-to-symbol mapping,
- Puncturer for ¾ rate coding, and
- STF, LTF, Header, Tail and Pad structure are similar in both

This standard is apt for very large scale process control applications like smart grid networks.

Communication Protocols

Communication protocols are an important aspect of WSNs as they form the foundation of communications between different components of a system. This can be between sensors, routers, and/or basestation. Following are the main ones for different WSN implementations:

1) **Bluetooth Low energy (BLE)**: In [32] a new strain of Bluetooth, capable of extremely low energy consumption with nominal data rates is outlined. It
consists of sleep periods unlike the classic Bluetooth which drops the duty cycle from 1.0% to 0.1%, and a greater modulation index, compared to the radios in previous versions thereby improving the coverage area. Moreover, this new version comes with greater security and extended battery life. However, interference with other devices might be an issue as the technology operates in the 2.4GHz ISM band. Additionally, it did not fit the requirement of mesh topology during the modelling of the presented system. Figure 2.8 shows the stack of BLE.

2) **LoRa**: is a proprietary protocol that uses a chirp spread spectrum (CSS) modulation technique [33]. It has extremely low current consumption of 10mA. The associated MAC layer of LoRa is open-source, and offers a full bidirectional symmetrical link between endpoint and gateway [34]. It also differentiates the MAC into three types i.e. baseline, beacon and continuous classes as shown in figure 2.9. LoRa is deployed for wide area networks (WANs) thereby acquiring the name LoRaWAN. It is useful for WSN systems that require long battery life and can be deployed at regional, national or global level. The network can be categorised into two categories: LoRaWAN public network, and LoRaWAN private network [34] depending upon the needs of an application.

Limited hardware support, and compatibility with other devices restrained in the use of this protocol for the presented system.
3) **SigFox**: is designed for long range communications between a client and a basestation. The network of SigFox is similar to a cellular network, but designed specifically for IoT devices [34]. SigFox uses the UNB on the physical layer. This allows it to have flexibility, and be more resistant towards interference [35]. In addition, SigFox follows a cloud-based approach for its network. Here information is passed to the backend server directly. However, SigFox is an IoT oriented network i.e. a sensor directly connected to the internet, which is not the model suitable for the presented system. Additionally, SigFox network may require subscription, and is not supported on wide number of platforms like ZigBee. Figure 2.10 shows the corresponding protocol stack of Sigfox.
4) **Z-Wave**: is designed for home automation systems by Zensys and validated by Z-Wave Alliance. Its main goal is to easily manage automated devices at homes. It is capable of running different devices like lights, TV, air conditioning, cooking, and security alarms [36]. The appliances in the home fall into two categories: controllers and slaves. The controllers can be primary or secondary and contain network information, but have different operations that can be found in [36]. The slaves receive and execute commands from the controller.

Z-Wave has an interesting feature called beaming, where the appliance switches off the radio for 1.5 seconds, and then turns it back on for 1.5 seconds. During the time, it is awake, the appliance searches for beam node frames. If discovered, only then the appliance stays awake continuously and begins other required network procedures. This feature allows for battery saving in nodes. The architecture of this protocol encompasses application, routing (Network), MAC, and transfer layers as shown in figure 2.11.

Z-wave is designed specifically for home automation, and had restricted features and interoperability issues until Z-wave’s components were released to public in September 2016.
5) **Thread**: is a networking protocol designed for smart home automation systems. It makes use of 6LoWPAN i.e. the tunnelling of IPv6 packets over the IEEE 802.15.4 wireless protocol. Thread is IP-accessible and allows for cloud access and Advanced Encryption Standard (AES) encryption. However, in the presented system direct IP access of the sensors was avoided, by using RIME communication stack, to prevent IP attacks directly on the sensor nodes. Additionally, Thread is a younger network than others and may not have as much support across different devices as ZigBee.

6) **TV White Spaces**: TV white spaces do not fall under protocols category but are worth mentioning because they are the unoccupied regions in the TV spectrum that have the potential to be used for WSNs. The reason for their usability is that the TV white spaces provide high bandwidth at long range transmissions [37]. Authors in [37] discuss a system that makes use of these white spaces for the benefit of WSNs while overcoming some of the associated challenges like energy efficiency and scalability. The advantages of using whitespaces over Zigbee/802.15.4 are:

- They offer wider number of channels (6MHz per channel)
- They have high availability, especially in urban areas.
- They can penetrate walls

Despite these advantages it was not viable to implement the system using the TV white spaces because:
- The band is not unlicensed everywhere
- Hardware may need new antenna
- WSN area with TV spectrum is still in its early stages of research

7) **ZigBee**: Based on the IEEE 802.15.4 standard, the ZigBee protocol defines the upper layers of the protocol stack and is optimised for systems demanding low energy consumption with low power and resulting low data rate. The low data rate is ideal for patient monitoring as not a large amount of data will be transferred for mundane tasks like temperature monitoring or blood saturation levels check. A drawback of this technology is low Quality of Service (QoS) and message throughput. Due to this drawback, the system integrates reliable secure routing, and cluster elections in Chapter 4 and Chapter 6 to make-up for the low QoS. ZigBee supports star, tree and mesh topologies. Although the use of mesh topology implies that it will not support super-frame structures and thus is incapable of sleep mode. Figure 2.12 shows the ZigBee stack coupled with the corresponding IEEE 802.15.4 standard.

![ZigBee Stack](image)

**Figure 2.12**: ZigBee communications protocol stack coupled with IEEE 802.15.4 stack. The top two layers are ZigBee while the lower two layers are 802.15.4

Table 2.1 and Table 2.2 show comparison between the protocols. Considering these metrics, and the details mentioned in the above points it can be deduced that different protocols are apt for communication in different applications and scenarios. For example, SigFox and LoRa are suitable for applications requiring long range communications over different geographical areas, this can be Structural monitoring, or monitoring the marine-bed environment and so on.
Z-Wave and THREAD are designed and suitable for home automation systems. ZigBee and BLE are suitable for communication between the wearable sensor nodes and the access points / smartphones. This is because of their nominal data rates, low latency and low energy consumption. Moreover, adaptive frequency hop spread spectrum allows BLE to co-exist with Wi-Fi, as the latter may be used for communication between base station
and Hospital main server. However, ZigBee is selected as the protocol for the presented system because:

- Out-of-box use with unlicensed band everywhere in the world
- Larger number of devices are supported
- Hardware is cheap
- Nominal data rates and low energy consumption.

<table>
<thead>
<tr>
<th>Metrics\Protocol</th>
<th>LoRa</th>
<th>SigFox</th>
<th>Z-Wave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>15-20 Km</td>
<td>30-50 Km (Countryside)</td>
<td>Upto 30 meters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-10 Km (Urban)</td>
<td></td>
</tr>
<tr>
<td>Power / Current</td>
<td>Extremely Low (10 μA)</td>
<td>--</td>
<td>1 mW (output) or 0 dBm</td>
</tr>
<tr>
<td>Topology</td>
<td>Star</td>
<td>Star</td>
<td>Source-routed mesh topology</td>
</tr>
<tr>
<td>Data Rate</td>
<td>0.25 – 50 Kbps</td>
<td>100 bps</td>
<td>9600 bps &amp; 40 kbps</td>
</tr>
<tr>
<td>Modulation</td>
<td>Chirp Spread Spectrum (CSS)</td>
<td>BPSK</td>
<td>FSK</td>
</tr>
<tr>
<td>Radio</td>
<td>868 MHz (EU), 915 MHz (US), 433 MHz (Asia)</td>
<td>868 MHz (EU), 915 MHz (US)</td>
<td>868 MHz (EU), 915 MHz (US)</td>
</tr>
<tr>
<td>Remarks</td>
<td>Wider band than SigFox (&gt; =125 kHz). Better bidirectional func than SigFox. [149]</td>
<td>Sophisticated BS. Cannot carry heavy data. Less interference than LoRA. [149]</td>
<td>--</td>
</tr>
</tbody>
</table>

### 2.1.4 Platforms

This section describes the hardware and simulator/emulator that may be used for different WSNs and their applications:

**Tmote-Sky**: is a legacy sensor node based on the MSP430 microprocessor but is still used for testing in WSNs, and beginners training, because of its compatibility with different software and vast online support. It has an 8MHz microprocessor with 10 kb RAM, and 48 kb of flash storage. In addition, it is supported by all the well-known
operating systems like Contiki OS, TinyOS, and Mantis OS among others. Figure 2.13\(^2\) shows the Tmote-Sky in detail.

![Tmote Sky. Figure shows hardware components](http://2.bp.blogspot.com/-P2lSfqrCdag/TolBndq6lTI/AAAAAAAAAN0/ezPvSAIHsQs/s1600/front.png)

**WaspMote**: is a commercial mote with the ATmega 1281 microprocessor, running at 14.7456 MHz frequency. It uses an XBee 802.15.4 Pro module having a radio range of up to 7 km. The modules provide 2GB external storage in the form of a micro-SD memory. These motes are useful for industrial deployments, and marine based WSNs \[^{18}\] due to their long range radios, and easy-to-use integrated libraries. Figure 2.14\(^3\) shows the WaspMote in detail.

![WaspMote](http://www.libelium.com/resources/images/content/products/waspmote/overview/hardware/hardware_top_small.png & https://www.cooking-hacks.com/media/cooking/images/documentation/article_waspmote/waspmote_ethernet_expansion_radio.png)

**Shimmer3**: is a medical based WSN node comprising of the TI MSP430 microprocessor running at 24MHz frequency with 16 kb RAM, and 256 kb of flash storage. It has several sensors built into it for e.g., gyro, low noise accelerometer, wide-range accelerometer, and a pressure sensor. Figure 2.15\(^4\) shows the Shimmer3 node in detail.

\(^2\) Source: http://2.bp.blogspot.com/-P2lSfqrCdag/TolBndq6lTI/AAAAAAAAAN0/ezPvSAIHsQs/s1600/front.png

\(^3\) Source: http://www.libelium.com/resources/images/content/products/waspmote/overview/hardware/hardware_top_small.png & https://www.cooking-hacks.com/media/cooking/images/documentation/article_waspmote/waspmote_ethernet_expansion_radio.png

\(^4\) Source: http://zeitgeistlab.ca/doc/doc_images/wban05.png
Openmote: is based on the CC2538 System-on-Chip (SoC) as shown in figure 2.16(a). The pins on the XBee breakout are mapped on to the pins of the CC2538 chip.
These XBee pins are used to connect to OpenBase as illustrated in figure 2.16(b). The chip provides superior features compared to its predecessors, such as: longer radio range, better partial configurations (low power modes) and a bootloader locking mechanism.

There are several other sensor nodes available in the market with different configurations and applications [38]. The ones mentioned here are the most common and popular wireless sensor nodes. Out of these, openmote is the node selected for our implementation of a secure medical based WSN system. This is because of its high storage capabilities, superior features such as: longer radio range, better partial configurations (low power modes) and a bootloader locking mechanism, and it supports the well-known Contiki OS, unlike the medical-based Shimmer3 nodes. The usage of this OS is advantageous for the reasons mentioned in the following section.

2.1.5 Operating Systems

Most sensor nodes, mentioned in Section 2.1.4, require the usage of an operating system (OS) for full functionality. An OS running on these nodes must fulfil the requirements of
1) reliability, 2) efficiency, and 3) have an adaptive communication stack. However, every OS does not fulfil all the requirements at all-times. Therefore, in this section the most popular operating systems are reviewed, and the justification for the selection is provided based on their advantages, disadvantages, and suitability to the presented system.

The most common and popular operating systems are:

**FreeRTOS**: is a Real-Time Operating System (RTOS) with soft and hard real-time requirements [39]. It is widely used for microcontrollers and small microprocessors. The operating system is simple with three source files common to RTOS ports (controller-specific files), and only one microcontroller specific source file [40].

FreeRTOS reduces code footprint by excluding unused functionality [39], and conserves power with a tick-less operation [40]. FreeRTOS provides coroutine and trace support with task, queue, resource, and memory management. In addition, the FreeRTOS scheduler provides both preemptive and cooperative options unlike most of the other operating systems.

FreeRTOS implements tasks as threads. Its application consists of several tasks. At the top level, each task may have one of the two states i.e. running(r) or not running(nr). The nr state can further be expanded into blocked, suspended, and ready states. The explanation of each state is out of the scope of this chapter, but can be found in [41]. The state of a task is governed only by the FreeRTOS scheduler, which may swap in (nr -> r), or swap out (r -> nr) a task depending upon the requirements of the application.

Despite the above mentioned advantages, FreeRTOS has its downfalls as mentioned in [42]. For instance,

- FreeRTOS has no one-shot task support.
- It has overhead in ISR execution times due to service calls.
- Its’ Kernel operations need protection from interrupts.
- FreeRTOS service failure reports do not give enough reasons (only asserts).
- FreeRTOS application Compilation results may be unpredictable in the release build when a task runs to the last braces `}` of main function.
- FreeRTOS does not come with a ZigBee stack integrated, which is very important for the WSN system presented in this thesis.

Therefore, for the reasons mentioned here FreeRTOS is deemed unsuitable for the implementation of our secure end-to-end medical-based WSN system.
**RiOT OS**: is a free, and open-source operating system implementing the relevant open standards for internet of things [43]. The roots of this operating system go back to the FeuerWare project in 2008. The latter was an OS for WSNs at that time. Working at the repository of FeuerWare, the people involved integrated $\mu$Kleos [43] with 6LoWPAN, **IPv6** Routing Protocol for Low-Power and Lossy Networks (RPL), and Transmission Control Protocol (TCP). Thereby giving rise to the current version of RiOT OS over the coming years.

RiOT OS has standard multi-threading capabilities, and comes with standard tools for C/C++ programming with one code for 8-bit, 16-bit and other platforms. It has a developer-friendly API [43], and allows dynamic memory allocation.

Some of the other features:

- The operating system’s kernel is robust against bugs as it has a modular microkernel architecture [43].
- RiOT OS has vast networking support for 6LoWPAN, IPv6, RPL, UDP, CoAP, and CBOR technologies.
- It also supports multiple utilities like Shell, and Secure Hash Algorithm (SHA-256).
- In addition, the minimum RAM and ROM footprint of RIOT OS is lower compared to Contiki, and Tiny OS [43].

Some features are fully supported like IPv6, UDP, TCP, 6LoWPAN and so on, however, some features are missing like HTTP support, and IEEE 802.15.4 MAC layer because it is still under development. The former is important for running HTTP web servers on the motes, and the latter is important in the designing of the secure routing protocol in [44] (Chapter 5). This is one of the reasons for not choosing RIOT OS. Another reason is that the high-capacity openmote [45] is used, which means that slightly higher RAM and ROM consumption is not an issue while running Contiki OS on them.

**Tiny OS** [46]: One of the first stable releases of operating systems for WSNs. This OS is flexible and application-specific, and is based on event-driven programming.

The main computation abstractions involving the OS are as follows:

1) **Commands** are a request that can be used to perform a service
2) **Events** are used to signal the completion of services.
3) **Tasks** are posted by commands and events to perform computation later-on. Thereby allowing them to return immediately.
CHAPTER 2. SENSOR NETWORKS BACKGROUND AND OVERVIEW

Figure 2.17 shows how the command request and event signal are de-coupled in TinyOS. This is advantageous because it allows the command and event to return immediately. Thereby, allowing other processing to continue, if required. However, the disadvantage is that the programmer needs to manually keep track of commands/requests and events/signals. This can get tedious when programming large applications. This is unlike Contiki OS where one need not keep track of commands and events manually. This is because in Contiki OS, the programmer only needs to look out for processes, which are implemented as event handlers [47]

TinyOS supports the nesC programming language (dialect of C) with an Event-driven and component-based programming approach. The OS enforces low power consumption due to split-phase operation and event-driven execution model. TinyOS also has low peripheral power consumption because it allows subsystems to be put in a low power idle state [48]. The OS is memory efficient with minimum RAM and ROM sizes equating to < 1KB and < 4KB respectively. Finally, the OS is small in size, and is capable of optimizing the code footprint (8 to 60 %) depending upon the application used [46].

Some of the disadvantages are:

- Risk of stack overflows need to be avoided since some MCUs supporting TinyOS do not have memory management units (MMUs) or any other memory protection measure [48].
- Programming is complex due to the event-driven approach for modelling an application [48].
**Contiki OS:** It is a lightweight and flexible OS built for resource constrained environment with an event-driven kernel, and pre-emptive multi-threading. It maintains a lightweight and compact system while using dynamic loading and unloading features [47].

Contiki OS allows programming in C/C++ with standardized tools, and easy compiling system for multiple platforms. The compiling system also allows dynamic program replacement at run-time [48].

The main features are:

- Platform independent kernel and service layer [48], with partial multi-threading capabilities.
- Power awareness mechanisms for development for low-power systems, with highly efficient radio duty cycling (RDC) layer.
- Full IP networking with 6LoWPAN, IPv6, RPL, UDP, TCP, HTTP, and CoAP.
- Cooja Network Simulator (Emulator) that is capable of emulating common hardware platforms.
- Multiple Cryptographic utilities like SHA-256, AES, and Elliptic Curve Cryptography (ECC).

Despite the advantages, the OS has some downfalls like the minimum RAM and ROM consumption is higher compared to other operating systems, Table 2.3 (at the end of this section). This is because the scheduler and kernel are complex, and the OS allows dynamic program-loading [48]. Additionally, the RAM consumption increases when the multi-threading library is included.

**System overview**

The Contiki OS consists of Kernel, Libraries, Program loader, and Processes. The system here is partitioned into core and loaded programs as shown in figure 2.18.
Kernel: The contiki kernel comprises of lightweight event scheduler, handles program execution, and provides polling mechanism. The kernel is associated with two types of events: synchronous and asynchronous. The latter is similar to events in TinyOS. Both type of events in contiki, are pre-empted by interrupts only, and no other events.

Communication Stack: The Contiki OS allows multiple stacks to be loaded because communication is implemented as a service. The communication stack of contiki is referred to as RIME [49]. Here most of the modules are based on the basic module called abc i.e. anonymous broadcast. This type of layered stack architecture helps in restricting the code footprint of the RIME stack to a range of 100 to 226 bytes only.

Hardware Abstraction Layer (HAL): Finally, Contiki has no hardware abstraction unlike TinyOS. All the hardware components are directly accessible to the contiki applications [48]. This was useful when building the secure routing protocol because the lower layers of the contiki stack i.e. radio duty cycling (RDC), and radio layers are directly accessible and were modified to suit the needs of the presented secure routing protocol in Chapter 4 (published in [44]).

All the above operating systems are presented in Table 2.3. In this table, a brief overview of each OS is given, and how it compares to the others.
Table 2.3: Operating Systems Specifications

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Contiki OS</th>
<th>FreeRTOS</th>
<th>RIOT OS</th>
<th>TinyOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>Layered</td>
<td>*</td>
<td>MicroKernel**</td>
<td>Monolithic</td>
</tr>
<tr>
<td>Scheduling</td>
<td>Event-driven (FIFO)</td>
<td>FPS¹ [41] / CS² / Hybrid Scheme</td>
<td>Circular Linked lists (Fixed size) [43]</td>
<td>Event-driven (FIFO)</td>
</tr>
<tr>
<td>Real-Time Support</td>
<td>Partial</td>
<td>Complete</td>
<td>Complete</td>
<td>None</td>
</tr>
<tr>
<td>Min RAM</td>
<td>&lt; 2KB</td>
<td>--</td>
<td>~ 1.5KB</td>
<td>&lt; 1KB</td>
</tr>
<tr>
<td>Min ROM</td>
<td>&lt; 30KB</td>
<td>6 – 12 KB (Kernel only)</td>
<td>~ 5KB</td>
<td>&lt; 4KB</td>
</tr>
<tr>
<td>Language</td>
<td>C dialect</td>
<td>C</td>
<td>C/C++</td>
<td>NesC</td>
</tr>
</tbody>
</table>

*Kernel is propriety. **Inherited from FireKernel
FPS¹: Fixed Priority Pre-emptive Scheduling. CS²: Cooperative Scheduling

2.2 Summary

The chapter detailed the concepts of Wireless sensor network and introduced existing systems for healthcare in Section 2.1.1. The concepts revolving around this field were studied in detail and compared to each other using various parameters. The wireless standards that were studied are IEEE 802.15.4, 802.15.6, and 802.15.4g. The corresponding communication protocols studied are BLE, LoRa, Sigfox, Z-Wave, THREAD, TV White spaces, and ZigBee. The choices made for the presented system in this thesis are IEEE 802.15.4 with Zigbee due to the reasons mentioned in Section 2.1.3.

This chapter also describes the hardware and software used to bring the system to life. The hardware platforms that were studied are: Tmote-Sky, Wasp mote, Shimmer3, and Openmote, while the software i.e. the operating systems studied were FreeRTOS, RIOT OS, TinyOS, and Contiki OS. Out of these, Openmote and Contiki OS were selected for the implementation of the presented system due to the reasons mentioned in Section 2.1.4 and Section 2.1.5.

This summarizes the background associated with WSNs systems, however there is one parameter not discussed in this chapter i.e. Security. This will be explained in detail, with the overview of the presented system, in the following Chapter (Chapter 3).
Chapter 3

SAS: System and Security

Progress in smart technology is going on for past few years with advancements in WSNs. These networks consist different types of sensors depending upon the applications for which the WSN is used, which is already covered in Chapter 2. In addition, the system also contains other useful components like routing nodes, sing/gateway, Internet, and an end-user device like a laptop/PC as shown in figure 3.1.

Each of these require different types of security measures. Therefore, this chapter discusses the security of the system keeping in mind the overview of the system, and its’ associated processes.

The rest of the chapter is structured as follows: Section 3.1 describes the system overview that encompasses system phases, security protocols and communication protocols. Section 3.2 reviews the security by categorising into different parameters for WSN and the implemented system. Finally, Section 3.3 summarizes the chapter.

3.1 System Overview

In this section the implemented end-to-end secure WSN solution is outlined. Figure 3.1 shows the full system from one end to the other. Here, the different phases of the system are shown namely: Initial setup phase (not in figure 3.1), Cluster Head (CH) elections, and Secure Routing.
1) Initial setup phase: where information is pre-embedded onto the sensors and/or other network devices. During this phase, it is important to note that the sensors/devices are not compromised otherwise the system will be compromised. Authors in [50] provide a mechanism to secure WSN against tampering, thereby reducing this issue.

2) CH Elections: In this phase the cluster of sensors is responsible for electing a cluster head based on the parameters and metrics mentioned in Chapter 4. The sensors in the cluster are responsible for sensing data, and electing a CH. This CH communicates with the router for forwarding data, and mediating security measures as shown in Chapter 4 and Chapter 6. This phase is important because it conserves energy of the network [51].

3) Secure Routing: phase allows for a secure multi-hop connection between the source Router (R), and the destination BaseStation (BS). This phase protects the routing connections against Denial of Service (DoS) attacks like black-hole and selective forwarding, and is explained in detail in Chapter 5. The phase is important because the multi-hop routing adds mobility to the system, thereby allowing patients in the hospital to be mobile; and the security makes the system more reliable to the users.

4) Internet: phase is responsible for connecting the gateway to the subscribed PC via a publish/subscribe model using Message Queueing Telemetry Transport (MQTT) communication protocol described in Section 3.1.2. This phase is simulated by the Mosquitto MQTT broker as shown in figure 3.2. The PC publishes commands to the broker, and subscribes to data. The gateway publishes data to the broker and subscribes to commands. The broker runs on a Linux PC and is configured via the /etc/mosquitto/mosquitto.conf file.

The above phases are integrated with different types of security measures that are described in their respective chapters. However, the outline of the security mechanisms implemented along with the communication protocols used by the system is described in the following section with reference to figure 3.2.
3.1.1 Security Protocols

Following are the security protocols and their usage:

1) ECDH [52]: Elliptic Curve Diffie-Hellman algorithm is used for generating and updating session keys between the router R, and the cluster of sensors as illustrated in figure 3.2. The algorithm, as the name suggests, makes use of elliptic curve cryptography (ECC). This protocol is used for public-key cryptography in digital signatures and/or key management. ECDH is a key-establishment protocol that is based on the Diffie-Hellman exchange.

   - First, both parties decide on an elliptic-curve $E$ either over finite field $F_p$ or $F_{2^m}$. The decision is based on many factors that like node processing power, capacity, and cryptographic relevance. Details for this and other elliptic curve parameters can be found in [53], [54].
   - Next a generator point $G(x,y)$ is selected. This is a point that lies on $E$, such that the order of $G$ is normally prime.
   - Bob, and Alice then generate their respective Public keys using their private keys and the generator point as shown in figure 3.3. Here, the private keys of Bob and Alice are $k_B$ and $k_A$ respectively. These private keys are generated from the pseudo random number generator mechanism that is shown in Section 5.4. Now, the public keys are formulated as shown in Eq. 3.1.

$$Q_B = k_B.G, \quad Q_A = k_A.G$$  \hspace{1cm} (3.1)

   - Bob and Alice send their public keys to each other ($Q_A, Q_B$).
Bob then generates session key $K_S = k_B.Q_A$, and Alice generates session key $K_S = k_A.Q_B$ as shown in Eq. 3.2.

$$K_S = k_A.Q_B = k_B.Q_A = k_A.k_B.G \text{ (from (3.1))}$$ (3.2)

Hence, the generated key on both sides is the same.

2) HKDF [55]: HMAC-based Key Derivation Function is used to update session keys between the subscribed PC, and the router R from the system figure previously presented in figure 3.2, where the HMAC refers to a Hash-based Message Authentication Code, which consists of two arguments: a key, and a message.

There are two phases to HKDF: Extract, and Expand.

**Step 1: Extract**
- HKDF-Extract (salt, IKM), where Input Key Material (IKM) is used as a message input in this stage.
- This results in a Pseudorandom Key (PRK) of HashLen octets, where HashLen is the length of the output of the Hashing scheme used.

**Step 2: Expand**
- Output Keying Material (OKM) = HKDF-Expand (PRK, info, L), where info is an optional context, and L is the length of OKM in octets and is less than or equal to HashLen*255.
- The calculation of OKM, and HKDF’s security strength is presented in detail in the HKDF Request For Comments (RFC): RFC5869 [55], and is implemented in the same fashion for the presented system in figure 3.2.
3) TLS [56]: Transport Layer Security is the latest in authentication based security mechanism. This algorithm is used to ensure that a server (mosquitto broker) and its clients (PC, and Gateway G) are authentic in figure 3.2. It provides communications security over a computer network.

4) AES GCM-256 [57]: 256-bit Advanced Encryption Standard, running in Galois Counter Mode, is used to ensure high standards of data confidentiality and data integrity. The reasons for selecting this mode over other AES modes like CBC are:
   - The GCM version provides one pass authentication with encryption.
   - High efficiency and performance because it’s a block cipher.
   - Can take advantage of parallel processing (in PC).

Figure 3.4 shows the inner workings of the GCM mode for AES.

![Figure 3.4: AES GCM mode of operation. Here, {E}k is the session key, and Mult (H) is the Galois multiplication function](image)

3.1.2 Communication Protocols

Following are the communication protocols used by the system:

1) RIME [49]: is a communication stack designed specifically for the Contiki operating system. It is a lightweight layered stack for wireless sensor networks. The main goal of the protocol is to allow communications between sensor networks while keeping the stack small, and flexible. It is a layered architecture as shown in figure 3.5.
Here, each layer is simple with a header of few bytes, and enables code re-use. The core memory footprint of rime ranges from 100 to 226 bytes only. This is extremely low because the processing is shifted from the applications to the system’s core. Thereby, reducing program energy consumption [49] [58]. For further evaluation and analysis of RIME, refer to [49].

The RIME communications used in the presented system is a mixture of the following primitives:

- **Broadcast**: module sends packet to all destinations that are in the radio’s range.
- **Unicast**: module sends packet to a destination that is a single hop away.
- **Polite**: module is anonymous best-effort local broadcast. For every time interval, it sends one local area broadcast packet. If another packet with same header is received during this interval, the packet is not sent.
- **Netflood**: module is used for best-effort flooding. The module sends a packet to all nodes in the network through flooding, and polite broadcasts. It uses END_TO_END sender, END_TO_END packet ID, and Time To Live (TTL).
- **Mesh**: module sends packets to a receiver in the network that is more than one hop away. The multi-hop mechanism is modified to suit the needs of secure routing protocol in Chapter 4.
- **Trickle**: module is a reliable single-source multi-hop flooding primitive. It sends a packet to all nodes of a network.
Apart from these primitives there are many other dependencies for the RIME stack, these can be found in the source code of contiki\(^5\).

2) MQTT [59]: is an extremely lightweight and standardised (OASIS) protocol used for communication between two machines (M2M) using a publish/subscribe model. The protocol has compact size, low power usage, and minimised data packets, and is therefore widely used for home automation, and medical (healthcare) applications. It is useful for communicating data from a WSN sensor to an end-device (say laptop or mobile) via a broker like mosquitto as shown in figure 3.6. The broker is the only one who implements the MQTT server-side protocol, all the other devices connected via the broker implement MQTT client-side protocol with publish and/or subscribe features.

![MQTT Broker](image)

Figure 3.6: MQTT Publish/Subscribe model. Gateway publishes data to topic WSNsystem/patient\(0\)/temperature, and the laptop subscribes to the same topic to receive humidity data. Conversely, the PC can publish command to topic WSNsystem/patient\(0\)/cmd, and the gateway receive this command by subscribing to the same topic

An important aspect of MQTT is its `topic`, which is a UTF-8 string used by the broker to sort messages for each connected client [60]. The format for a topic is as follows:

\[
\text{UnivOfLim} / \text{mainbuilding} / \text{secondfloor} / \text{humidity} \tag{3.3}
\]

Each of the phrases in Eq. 3.3 refer to a topic level, and each topic level is separated by a forward slash. A topic starting with a `$` is reserved for the internal workings of a broker, and a topic may contain different wildcards like `+` (single-level) or `#` (multi-level). A Single-level wildcard example would be:

\[
\text{UnivOfLim} / \text{mainbuilding} / + / \text{humidity} \tag{3.4}
\]

In Eq. 3.4 the single-level wildcard implies that the broker must accept humidity data to/from any floor on the mainbuilding.

\(^5\) https://github.com/contiki-os/contiki
A Multi-level wildcard example would be:

\[
\text{UnivOfLim} / \text{mainbuilding} / \# \quad (3.5)
\]

In Eq. 3.5 the multi-level wildcard implies that the broker must accept any data to/from any floor on the \textit{mainbuilding}.

From these examples, it is seen that it is important to get the topic and its concepts right otherwise the data published may be going into oblivion.

Now the above features are rendered useless if they are modified by an attacker for their personal gain. Therefore, security is an important requirement in conjunction with this MQTT protocol. It has a few features that can improve the security of a connection. First, the new version v3.1 provides the usage of username and password, which is useful for source/destination authentication. Second, it provides encryption-authentication across the network with assistance from secure socket layer (SSL). However, the addition of SSL will incur significant network overhead [61]. In addition, programmers may encrypt and/or hash the application data to provide confidentiality and/or integrity manually.

These features have been integrated in the presented system and are explained in Chapter 5.

3) UART (Serial): microchip allows a computer/smart-device to interface with the attached serial devices. It allows a device to talk to and exchange data with other serial devices as shown in figure 3.7.

![UART Diagram](image)

Figure 3.7: UART on a computer/gateway connected to a serial device (say openmote)

### 3.2 Security Overview

#### 3.2.1 Security Services

Confidentiality refers to the protection of data from unauthorized users or malicious individuals, the lack of which can have serious repercussions on the smooth working of a system. In the system presented here, confidentiality is provided by making use of the AES - Galois/Counter Mode (GCM) [57] with 256-bit security. This mode has several advantages over other cipher modes [62], such as:
1) Accepts arbitrary length initial-vector (IV).
2) Provides one-pass data authentication.
3) Can be used as a stand-alone MAC or incremental MAC.

The authors in [62] show how the GCM mode is secure after using the above mentioned features. Thereby, increasing our confidence in the usage of this mode for the presented system.

Integrity refers to the protection of data from any modification. To provide this, the system uses the previously mentioned AES-GCM 256 (Section 3.1.1). This approach is useful as it allows a one-pass data authentication with confidentiality.

Authentication provided refers to the authorisation of an entity that wishes to join the network. Also, allowing a network device to verify the authenticity of a data/packet sender. This is achieved in two parts:

1) BS to router R: A SHA-256 hash-based scheme is implemented. This scheme helps the router verify the authenticity of the BS, [44].
2) PC to the BS via Gateway: A preloaded-key $K_m$ is used with a randomly generated salt $S$. Additionally the TLS protocol, mentioned in Section 3.1.1 and Section 3.2.6, provides authentication for the Internet side of the system between the PC and the Gateway to prevent tampering, message forgery and eavesdropping [56]. This protocol runs on port 8883 and authenticates the gateway and the PC via X.509 certificates.

### 3.2.2 Security Attacks

Security attacks have an adverse effect on WSNs. These attacks can wreak havoc on a system by denying user service, stealing private data, and compromising individual nodes thus leading to system problems. Security attacks can be classified in various forms. They can be classified as either active or passive, or as per the different layers of the open system interconnection (OSI) model that gives rise to physical security (Physical Layer), routing security (Network Layer), and cloud-side security (Transport Layer)

#### Passive vs Active attacks

1) Passive attacks: In this attack an adversary monitors the traffic in-order to gain information about the network or user or the service, however, the adversary does not interfere with the on-going transmission(s) and/or the message(s). Example of this type of attack: traffic monitoring, which is used to prepare for a future active attack. This type of attack can be harder to detect than active attacks as the adversary does not intend to arouse suspicion.
2) Active attacks: are more intrusive than passive attacks. This is because the adversary launching this attack manipulates the on-going traffic to gain information and/or access the system under attack. This type of attacks can cause more damage. Some of the examples are: spoofing, jamming, selective-forwarding, Sybil attack, HELLO Flood attack and wormhole attacks. Some of these attacks may be harder to prevent due to the complexity involved. Most of these active attacks are described in Section 3.2.5.

Attacks classified on OSI layers

Table 3.1 shows common security attacks on the different layers of the OSI stack.

Table 3.1: Security attacks based on layers of the OSI model

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>Security Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport layer</td>
<td>Flooding attack</td>
</tr>
<tr>
<td></td>
<td>De-synchronisation attack</td>
</tr>
<tr>
<td>Network Layer</td>
<td>Blackhole attack / Selective Forwarding</td>
</tr>
<tr>
<td></td>
<td>Sinkhole attack</td>
</tr>
<tr>
<td></td>
<td>Sybil attack</td>
</tr>
<tr>
<td></td>
<td>ACK Spoofing attack</td>
</tr>
<tr>
<td></td>
<td>Replay attack</td>
</tr>
<tr>
<td></td>
<td>Wormhole attack</td>
</tr>
<tr>
<td></td>
<td>Node Subversion attack</td>
</tr>
<tr>
<td>Data Link Layer (MAC Layer)</td>
<td>Exhaustion of Resources</td>
</tr>
<tr>
<td></td>
<td>Unfairness</td>
</tr>
<tr>
<td></td>
<td>Collision</td>
</tr>
<tr>
<td>Physical Layer</td>
<td>Radio Jamming</td>
</tr>
<tr>
<td></td>
<td>Side-Channel attack*</td>
</tr>
<tr>
<td></td>
<td>Node Tampering / Capture attack</td>
</tr>
<tr>
<td></td>
<td>Node Replication</td>
</tr>
<tr>
<td></td>
<td>Eavesdropping*</td>
</tr>
</tbody>
</table>

*Passive attacks

3.2.3 Physical Security

WSNs are susceptible to physical attacks. These attacks target the physical access to sensor nodes in a network or the physical layer of the OSI stack [63]. Some of the attacks are as following:

1) Node Tampering attack: allows an adversary to access a sensor node physically. This allows the adversary to control the node for their own benefit as they have access to node’s memory and the stored data. To protect against this attack one
can disable Joint Test Action Group (JTAG) access or use the interrupt vector randomization technique to replace the original interrupt vector table with 16 unused random code addresses [50].

2) Node Replication attack: is an extension of the node tampering attack. Here, after tampering a sensor node, the adversary uses information gained, from the node, to replicate it and insert themselves into the WSN as a legitimate node. To protect against this attack one may use the measures deployed to prevent node tampering in conjunction with the strategies mentioned in [64].

3) Node Malfunctioning attack: can create a simple jammer node by making the node generate inaccurate / inconsistent data thereby violating sensor node integrity and possibly sensor network data freshness. To protect against this attack one must provide strong authentication, authorisation and access control in conjunction with restriction on the number of nodes’ neighbours [65].

4) Side-Channel attack: allows an adversary to collect private information by monitoring the power consumption or the electromagnetic (EM) emanation of the sensor nodes. This leads to the adversary being able to launch further attacks using the acquired private cryptographic keys [63]. To protect against this attack one can deploy techniques like code obfuscation, and process obfuscation with tamper-proofing physical access to the sensor node(s) [66].

5) Jamming attack: is achieved by sending radio signals over the channel or by redirecting sent packets to sender transmitter instead of the receiver [67]. Some of the countermeasures against constant jammers are channel surfing, and spatial retreat. Other measures include reactive anti-jamming technique using wormholes [68].

6) Eavesdropping: can be either passive or active. The former refers to the scenario where a compromised node listens network transmissions to acquire information. The latter refers to the scenario where the compromised node grabs information by interacting with transmitters by pretending to be a legitimate node [69]. However, the active eavesdropping attack is a combination of passive eavesdropping and address spoofing.

3.2.4 Link Layer Security
Data Link Layer encompassing the medium access (MAC) layer is a crucial part of any network model. This is because it transfers data between adjacent network nodes. It is
responsible for detecting and correcting collisions that may occur when nodes try to use the same medium simultaneously.

The link layer can be a victim to a variety of attacks that include DoS attacks, and Collision:

1) Resource Exhaustion: is a type of DoS attack where a compromised node sends a series of packets to the victim node in order to deplete it off its’ battery resources.

2) Collision: refers to when multiple nodes send frames through the same channel simultaneously. This may lead to loss of data and hence denial of service.

There are different libraries provided with different operating systems that tackle some or all of the attacks on the link layer, these are TinySec [70], ContikiSec [71], and MiniSec [72]. Additionally, there are independent protocols that may be used like Link Layer Security Protocol (LLSP) [73], SPINS [74], C-Sec [75], and Steganography in MAC layer of IEEE 802.15.4 protocol [76]. Details of each can be found in [77].

In the presented system, the link layer of the contiki OS is used as is, but the radio duty cycling (RDC) layer is modified to suit the needs of the secure routing protocol in Chapter 5. This layer sits on top of the MAC layer (which facilitates CSMA protocol), and is responsible for, as the name suggests, duty cycling the radio. In the RDC, neighbourhood addresses are added to the acknowledgement (ACK) frame, and ACK capture + DATA capture functions are implemented additionally, which help in monitoring a node’s neighbour. The whole process is published in [44], and reviewed in Chapter 5.

3.2.5 Routing Security

Routing is an important part of a wireless sensor network because it allows for exchange of messages between parties that may not be in range with each other and/or not even on the same network. There are several types of routing algorithms including Distance vector algorithms, link-state algorithms, optimised link state routing, and path vector protocol. The protocol utilized in the presented system in Chapter 5 is an Ad-hoc On-demand Distance Vector (AODV) routing protocol. Since different types of routing protocols may have different security measures applied to them the security associated to the AODV-based routing protocol is discussed in Chapter 5.

46
Due to the open wireless nature of sensor networks, and the simplicity of its routing protocols, these networks are susceptible to most network layer attacks like the following [78]:

1) Replay attack: An adversary may capture information/messages in transit, and replay them at a later stage for their own benefit. The message, in addition, may be altered or spoofed. This attack is prevented in the presented system with the use of timestamps, and SHA-256 hashing algorithm (see Section 4.2.3).

2) Black hole: Since the AODV based protocol is dynamic in nature, an attacker can listen to the initial request packets (RREQ) and reply with a forged packet ‘request reply’ (RREP). This inserts the attacker in the routing path, and the attacker then drops any packet, thereby denying service to the destination/user. Therefore, this is a type of Denial of Service (DoS) attack that is prevented using the mechanism mentioned in Chapter 4, which is published in [44]

3) Selective forwarding: This is a specialized version of the black-hole attack. Here an adversary may compromise single or multiple nodes, and later forwards only few packets to the destination. This makes the compromised node less suspicious. The scheme represented in Chapter 4, [44] tackles this problem for the presented system.

4) Sybil attack: where a single node represents multiple identities on a network. This attack is more effective on geographic routing, and location aware routing protocols.

5) Sinkhole attack: here the adversary lures traffic from one area via a compromised node, thereby creating a sinkhole with the adversary in the middle. Sinkhole attacks can result in hidden selective forwarding attacks.

6) Wormholes: here an adversary convinces the sensor nodes that they are one or two hops away from the BS only, even if they are several hops away. This makes distant nodes believe that they are neighbours, which leads to them sending packets into oblivion. This attack is very difficult to defend as the solution requires clock synchronisation and accurate location verification.

7) HELLO flood attacks: takes advantage of the initial ‘Hello’ packet broadcast by nodes for announcing themselves. A laptop-class attacker may falsely broadcast packets thereby making other nodes think that the attacker is nearby. Thus, depleting their resources and Denying service to the system (DoS). This attack
is prevented, to some extent, in our system because the source of hello packet i.e. Router R4 is authenticated using ECDH in Chapter 5.

8) **ACK spoof attacks**: occurs in routing algorithms where implicit or explicit link layer acknowledgements are required. In a broadcast medium, an attacker spoofs the ACK of another node, to reply to the source node. Thereby, making the source believe that the destination is alive even when it is not.

### 3.2.6 Internet Security

It is important to secure connections of the WSN to the internet. This is because data could be travelling through multiple servers/access points across the internet, thereby leaving the network open to a wide range of attacks. Since internet security is a vast topic, its discussion in this section will be limited to the Message Queue Telemetry Transport (MQTT) protocol [79], and end-point devices i.e., Intel Edison, PC, and mosquitto broker (an MQTT Server) [80].

1) **MQTT Protocol**: The MQTT protocol is a light-weight flexible publish/subscribe model protocol as described in Section 3.1.2. The protocol does not have any solid security measures. However, it does allow the usage of *username* and *password* for connection between clients and server(s). This shows that MQTT can provide basic authentication for end-point devices in a cloud network. In addition, MQTT allows the usage of Transport Layer Security (TLS) for communications security over the internet. It provides secure communications for client/server applications by preventing eavesdropping, tampering, and/or message forgery [81]. Here, TLS can provide the programmer with usage of ciphers like AES, and RSA for confidentiality.

The inclusion of TLS has a drawback that it incurs overhead. This can be problematic if the MQTT is used for constrained devices. However, since the presented system in Figure. 3.1. uses MQTT on a Gateway (intel Edison) and a PC, therefore it is not a major issue as these devices have continuous wall-socket power supply. But, in case TLS is used on constrained devices one must consider the inclusion of Session Resumption technique to decrease the TLS overhead [82].

Finally, it is good practise for MQTT clients to validate the X509 certificate provided by the MQTT broker (*mosquitto broker* in this case) to prevent man-in-the-middle attack. More details about this implementation are provided in Chapter 6.
2) End-point devices: The end-point devices in the presented system (figure 3.1) are the Intel Edison Gateway (G) and the subscribed PC. These devices are described in detail in Chapter 2. They generate certificates that are used with the TLS protocol to authenticate themselves with the broker and vice-versa. The certificates are signed by a self-signed CA for testing purposes. This is because it is not possible to get the private key of a public CA. The handshake following this process is shown in Figure 3.8. Both the clients connect to the broker after certificates are validated. They can then publish/subscribe according to their needs.

![TLS Client/Server Handshake between Intel Edison, and Mosquitto Broker; followed by Linux PC, and Mosquitto Broker](image)

**RSC: Request Server Certificate**  
**RCC: Request Client Certificate**

3) Mosquitto Broker: The mosquitto server is an open source message broker implementing the MQTT protocol. The broker can be configured manually for providing different configurations and/or features to improve security and functionality. This is done using the file `/etc/mosquitto/mosquitto.conf` (for Ubuntu Linux OS)\(^6\). Figure 3.6 already presented the working of the publish/subscribe model with a broker in detail.

\(^6\) [www.mosquitto.org](http://www.mosquitto.org)
3.3 Summary

This chapter details the presented system, its components and different processes showing the inner workings of a WSN system with a flavour of components from the presented system. Each component was reviewed and analysed from security point of view. Common security attacks on WSNs were outlined, and countermeasures were provided. The attacks were also categorised based on different parameters like passive vs. active or attacks based on the different layers of the OSI model i.e. physical layer, data link layer, routing layer, and transport layer. Finally, the security attacks and countermeasures for the internet side of the WSN system were presented. Thereby, summarising the whole system, and its security briefly.

This forms the foundation of the system that will be useful in understanding the coming chapters beginning with the most important phase of the system i.e. Secure Routing.
Chapter 4

Secure Routing

The chapter introduces and discusses the main phase of the system i.e. Secure Routing. To assist with the explanations of this phase, figure 4.1 re-illustrates the system model for a medical WSN system. Here, the system comprises of different components: Sensors (S), nodes, Cluster head (CH), Routers and base station (BS). The sensors sample raw data, and send the values to the nodes which may convert data, and encrypt it for security. Following this, the nodes elect a cluster head, as explained in Chapter 6, the CH aggregates the encrypted data and sends it to the nearest router, which in turn sends it to the BS through a secure multi-hop routing process. To facilitate this, the system defines a routing protocol that can carry the data from source to destination by using either static routes or dynamic routes. The former is apt for static networks while the latter is good for mobile networks, namely, medical based wireless sensor networks.

![System Model](image)

Now the routing must be reliable enough so that data always reaches the BS. This is achieved by using a mesh routing protocol as it provides reliability through path redundancy. In addition, the protocol must be secure against common routing attacks.

Problem Statement
As illustrated in figure 4.1 data encrypted with AES-GCM 256 passes through the routing nodes. At this phase this data must be routed securely to its destination. This is because
in a medical based WSN the most important requirement is the safe delivery of patient-data to the end-user i.e. a doctor/hospital staff running the subscribed PC. If this information is mishandled it could put the system and the patient at risk. Hence, it is a requirement to secure the system against these attacks. These attacks fall under a class of attack known as Denial of Service (DoS) attacks. DoS attacks encompass a wide array of specialized or generic attacks that may be host-based or network-based. In this work the focus is on two DoS attacks i.e. Black-hole and Selective Forwarding (SF). Interested readers may refer to the overview on DoS attacks, methodologies and countermeasures in [83].

The remainder of this chapter is structured as follows: Section 4.1 presents current solutions for detection and correction of the above-mentioned attacks. Section 4.2 discusses the system setup with introduction to routing protocol in Section 4.2.1, Black-hole attack and consequences in 4.2.2, black hole countermeasure in 4.2.3, Selective forwarding attack and consequences in 4.2.4, selective forwarding countermeasure in 4.2.5, packet gathering in 4.2.6, and BS analysis in 4.2.7. Section 4.3 outlines Contiki OS protocol-modifications for achieving a defence against black hole and SF attacks. Section 4.4 measures results for an implementation of the modified and original protocols on Tmote-sky [84] and openmote [45] platform. Section 4.5 briefly summarises the security provided by the protocol. Section 4.6 summarises the presented phase of the system with emphasis on the results.

4.1 Current Solutions

In [85], the authors propose two different solutions to the problem of black holes in wireless sensor networks (WSNs). The first solution uses the multi-path property of redundant networks. The sender node waits for multiple response (RREP) packets to arrive and then decides which path is safer by eliminating the unsuitable path. The second solution requires sequence numbers. This is to keep track of the last packet ‘sent to’ and ‘received from’ every node.

The problem with the first solution is that it requires multiple shared hops between source and destination nodes. This hopping can incur time delays if there are not enough paths available in the network. On the other hand, the second solution incurs lower time delay but has no way of re-sequencing the packets received at the receiver. This may lead to garbled transmissions. Moreover, for both solutions the authors did not use any attack nodes that could simulate the attacks, and packet verification was done by verifying the routes only, which could result in false positives. Finally, they did not address the problem...
associated with a collaborative black hole attack, which involves the collaboration between multiple malicious nodes to bring down a network, as is the usual scenario in real world implementations.

In [86], the authors used threshold values and neighbouring promiscuous nodes to detect a black hole node. The issue with this solution is that each node must hold extra routing tables thereby leading to storage overhead.

In [87], the authors describe their scheme as being capable of defending against single and collaborative black hole attacks. They state that it is successful even when the node is idle. However, the scheme has separate mechanisms for idle and communicating nodes. When compared to the AODV protocol [88], the end to end delay increases by 100ms to 300ms for different number of attacker nodes i.e. 1, 4 and 5. The scheme also does not consider the grey-hole (Selective Forwarding) attacks for future work, where the malicious nodes drop packets selectively to avoid suspicion.

In [89] the authors simulate a multipath routing mechanism on the Omnet++ simulator. In this mechanism, they improve the reliability of the network by modifying the routing protocol. Here a node overhears the transmission of neighbour nodes. If there is dropping of packets, the node chooses an alternative path. This results in a reduced packet drop ratio. However, there is no mechanism proposed to isolate the malicious node causing the problem.

In [90], the authors combine the geographic routing [91], and watermark based schemes [92] to achieve high efficiency against selective forwarding attacks. The former allows the system to select a secure routing path while the latter can find and isolate the malicious nodes one at a time. However, the malicious node detection mechanism consumes higher energy than normal mode and there is transport delay due to the watermark extraction process. Moreover, it may be possible that a node becomes compromised after the process of selecting a secure route, thus falsifying the normal packet loss ratio. This could lead to undetected selective forwarding attacks in the later stages of the system.

The scheme presented in [93] is suitable for tree based protocols. The authors use watchdog nodes to monitor communication links between nodes. The watchdogs count packets in each time window, and the cluster head collects data from these watch dogs and then analyses it using probability theory. However, it is difficult to scale this architecture to large networks. This is because the required number of watch dog nodes will be higher and the traffic will be denser.
The proposal in [94] adds security to multipath routing. This is done by using MD5 authentication and encryption for confirming identities and securing multipath respectively. However, the use of encryption in routing can result in unnecessary overhead even before data transmission. Moreover, the authors do not provide any simulations or results to support the underlying mechanisms and algorithms.

Most of the schemes mentioned above either consume considerable higher energy than the original system or they are unable to isolate the malicious node. The secure routing presented here achieves a minimal gap ranging 141-293μW between the power consumption of the original protocol and the secure protocol as shown in Section 4.4.3. It also provides the ability to isolate the malicious node(s) and form a new path that is free of any malicious node. The work presented in this chapter provides both simulation results (Tmote-sky platform) and hardware test bed results using the openmote platform. However, the current solutions mentioned above prove their methods mostly using simulations only. This shows the superiority of the experiments conducted in this chapter as they allow real-world test-bed results to be obtained and verified in conjunction with simulation based measurements.

4.2 Experimental set-up

This section describes the preliminaries required to conduct the simulation, and builds towards setting-up a test-bed environment for hardware implementation.

![Network Model](image)

**Figure 4.2:** Network Model. Router 4 handles the cluster of sensors represented on the patient’s body.

The system model in figure 4.2 is the routing part of the network extracted and expanded from figure 3.1. It is based upon the sink-collection model presented in [95]. Router 4 collects data from its cluster of sensors represented on patient’s body, and periodically routes this data towards the base station (BS). This period is set by the PC application that is shown in Section 7.1, and is received by the router(s) during the route set-up phase that is explained in Section 4.2.1.
4.2.1 Routing

For routing, the network uses an on-demand routing protocol similar to the AODV protocol [96], which is made for a mesh network as the latter provides reliability through node redundancy [97].

This routing protocol is dynamic in nature, allowing the system to organise and heal itself for facilitating patients’ movements. It is self-organised because it uses an on-demand approach for setting up a routing path, and self-healing because it checks for breaks in the routing path due to the patients’ movements or node failure.

Now the routing protocol is modified to suit the needs of the presented system. The first modification is the reversal of the source of route request (RREQ) and route reply (RREP) packets, i.e., the destination node (BS) sends RREQ packet and the source node (Ni) replies with a RREP packet as shown in figure 4.3.

This modification has the following advantages:

1) Reduced network traffic compared to original AODV protocol as an attacker cannot flood the network with bogus RREQ packets. Thus, reducing chances of DoS attack.

2) Controlled network by the base station, which can set the data intervals for the routers on request from Subscribed PC (Figure 3.1). Thus, an attacker is incapable of changing the data interval, thereby reducing further possibility of a DoS attack.
4.2.2  Black-hole attack and consequences

Now, in a medical WSN system, using the routing protocol mentioned in section 4.2.1, an attacker can listen to route request (RREQ) packets and reply with a forged route reply (RREP) packet that advertises the shortest path to the destination. This creates a connection between source node and the compromised node. Thereby every packet is dropped by the compromised node thus creating a black hole. This process is called a black-hole attack [98], and is a cause of concern for the presented medical WSN system because:

1) The medical monitoring system will be at the mercy of the malicious node(s). Thus, sabotaging the main goal of the system i.e. patient-data delivery.

2) Physical security of the patients can be compromised as the patient’s vitals can be read and/or altered by a compromised node(s).

3) Workload of the medical staff may be increased when a compromised node triggers false alarms.

To get a better perspective of the attack, a simulation was run using the contiki (cooja simulator [99]). This simulation consisted of four nodes: one base station, two normal nodes and one malicious node (Node ID: 5) as shown in figure 4.4a. Here, node ID ‘5’ denies service to the network by forging a RREP packet. The malicious node ‘5’ then sends a forged RREP to node ‘2’ (see underlined text) from figure 4.4b. Following this, node ‘2’ sends data packets to node ‘1’ via node ‘5’. However, these packets never reach their destination because the actual path should be ‘2’ -> ‘4’ -> ‘1’.

![Figure 4.4: Black hole attack simulation on Cooja simulator (Contiki): (a) Nodes Layout. Source node green ‘2’, destination is blue ‘1’ and malicious node is red ‘5’, (b) Mote output: node ‘2’ sending data packets to node ‘1’ via node ‘5’. But these packets never reach their destination.](image-url)
From this simulation, it is inferred that a black-hole attack can make a WSN system useless. Therefore, a countermeasure for this attack is necessary, and is outlined in the following section.

4.2.3 Black-hole Countermeasure

The section discusses the countermeasure deployed for defending the presented system against black-hole attack. To proceed further a necessary assumption is that the base station (BS) is trustworthy. This is a basic requirement to facilitate a decision whether a node is malicious or normal as it rules out BS involvement in the attack.

The method to determine if a node is malicious or not is divided into two phases i.e. pre-deployment phase and routing phase:

*Pre-deployment phase:* Figure 4.5 illustrates the initial phase of the system. During this phase, every router (*circled numbers*) is in-range with its base station (BS). This is done to facilitate the distribution of unique random numbers from the BS to routers. The numbers are generated using contiki’s pseudo random number generator (PRNG) and a custom seed generator. Where, the latter uses the unique node ID as an initial seed to generate a final seed [100]. The usage of this random number is explained in the *routing phase* below.

![Figure 4.5: Pre-Deployment process: The routers (*circled numbers*) receive their respective unique random numbers from the PRNG running on BS](image)

After the generation of numbers, the BS distributes them using unicast messages. Here, it is assumed that this is a controlled delivery and that there are no malicious nodes listening to these messages. Following this, the routers are placed in their respective positions of the network like the one that was presented in figure 4.2 so that the routing phase can begin.

*Routing phase:* The routing process starts when the BS requires data, this may be the case when an external request by the subscribed PC via a cloud network is made. When the
BS recognizes that it needs patient data, it sends a request to the source node $N_i$ using the RREQ packet. The structure of which is shown in figure 4.6.

<table>
<thead>
<tr>
<th>destination</th>
<th>RREQ_ID</th>
<th>pad</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$Q = H(r_i \parallel TS)$

![Figure 4.6: RREQ Packet](image)

As shown the RREQ packet consists of a field named $Q$, where $Q = H(r_i \parallel TS)$. Here, $H$ refers to the SHA-256 hash algorithm, $r_i$ is the random number for the node, and $TS$ is the timestamp. The reason for this modification is that it provides source authentication, and prevents replay attacks. The former implies that the node $N_i$ is assured that the RREQ packet originated at the BS only. While the latter implies that an adversary cannot replay old messages to facilitate penetrating the routing path. The step is shown as follows:

1) The BS sends $(RREQ \parallel Q \parallel TS)$ to $N_i$.
2) When the source $N_i$ receives this route request message, it first checks for replay attack by verifying the timestamp:
   a. $N_i$ calculates $TS_d = TS_{current} - TS_{received}$.
   b. If $TS_d > \text{Threshold}$ (set by analysing simulation data) then reject the request and wait for another one.
   c. Else calculate the hash $Q'$ in the same way $Q$ was calculated.
3) Once the hash $Q'$ is verified by source $N_i$, it then:
   a. Sets the interval that is embedded in the request packet, figure 4.6,
   b. Sends back the reply packet (RREP) to the BS, figure 4.3c, and
   c. Finally, starts sending periodic data to the BS, figure 4.3b.

From the above steps, it is safe to say that a RREQ packet cannot be forged because of the inclusion of hash. Therefore, a malicious node cannot request data from the normal nodes in its pursuit of a black-hole attack.

In addition, there is no inclusion of a hash in the RREP packet. This is because it would be fruitless for a malicious node to forge a RREP packet, and start sending data to the BS. Since it is understood that the objective of a malicious node is to deny data from reaching the BS. The advantage of this exclusion is that it saves unnecessary processing at the nodes’ side, thereby consuming less energy.
4.2.4 Selective Forwarding attack and consequences

A selective forwarding attack is a specialised form of the previously mentioned black-hole attack, where [101], an attacker captures and compromises single or multiple nodes on the routing path. These nodes then drop packets selectively thereby avoiding suspicion. The attack is classified as single selective forwarding and collaborative selective forwarding respectively.

In this attack, malicious node(s) can maintain their cover by dropping packets selectively because wireless networks are lossy by nature. So, due to packet losses from interference and from the SF attack, it can be difficult to distinguish the reason behind packet loss here. The reason these packet losses may be harmful for the presented system are as follows:

1) Incomplete information, reaching the destination can be more dangerous than no information. This is because in medical terms one may not see the complete picture without all the data.

2) Altered patient information reaching the destination could result in incorrect treatment of the patient, thus physically and/or mentally harming them.

To safeguard the system against the attack a mechanism has been devised that helps protect the system against single and collaborative SF attacks. The following section describes this mechanism in detail.

4.2.5 Selective Forwarding Countermeasure

In this section the mechanism for defending against selective forwarding attacks is detailed. Here it should be noted that the BS and the data sending source node (N_i) are trusted. These entities are trusted because the RREQ packet cannot be forged as per the hash based scheme presented in Section 4.2.3.
The details of this mechanism are divided into five sections as illustrated in figure 4.7. These are: Neighbour Monitoring, Attack Detection, Control Packet Collection, Analysis and New Path Formation.

**Neighbour Monitoring:** Since the routing protocol is on-demand that means that every node has knowledge about its immediate neighbour only. So, for the system to detect the malicious node(s), the BS needs to have information on all the network nodes. To achieve this objective, every node \((i)\) is promiscuous i.e. they monitor data sent by their next node \((i+1)\) and Acknowledgement sent by their previous node \((i-1)\) as seen in figure 4.8.

This promiscuity with monitoring is achieved by modifying the radio duty cycling (RDC) layer of the Contiki OS as described in Section 4.3. The layer monitors packets only if a node falls on the route responsible for sending data from source to BS. This is required to avoid other nodes from monitoring packets unnecessarily and wasting processing power.

Now, the information generated from this monitoring of neighbours is formed into a packet called Control Packet (CP). Each node sends this packet towards the BS, when requested, and the BS uses these CPs to analyse the information for identifying malicious node(s).
**Attack Detection:** As seen above the CPs are analysed at the BS and not at every node. The reason for this centralized approach is that it does not give unnecessary authority to routers or other nodes.

In this process a unique sequence number embedded in the periodic data packet is used. Figure 4.9a shows data packets containing current sequence number and the next sequence number. Where the latter refers to the sequence number of the next data packet.

![Figure 4.9: Packets: (a) Data Packet, (b) ACK Packet](image)

The BS will acquire the first sequence number during the on-demand route formation phase because the source router node sends the first random Seq. No. attached with the RREP packet as shown in figure 4.10. This allows the BS to know the sequence number of the first ‘data’ packet that will be arriving soon. Following this, every data packet carries the Seq. No. of the next packet as illustrated in figure 4.10.

![Figure 4.10: Sequence Numbers in packets used for attack detection](image)

If the Seq. Nos. do not match at the BS, the BS increments a count for the number of packets dropped. If the packet-drop count increases beyond a certain threshold, then the BS deduces that the system is under a selective forwarding attack. Here the threshold
adjusts dynamically over time depending upon number of packets received. This process is supported by the assumption that the packets do not arrive out-of-sequence.

The usage of these sequence numbers is advantageous because:

1) They introduce randomness in the sequencing (through a PRNG-Pseudo Random Number Generator),
2) Allow the BS to keep packet count, and
3) Prevent *Replay attack* if the attacker does not know or have access to the sequencing algorithm.

Following this phase, the BS requests control packets (CPs) from the network nodes. *Control Packet collection:* Figure 4.11 illustrates the collection of the control packets at the BS. The source unicasts the first CP (CP1) to its next hop. If the next hop does not forward the message within a certain amount of time (*set arbitrarily*), then the source floods the network with CP1. When another node (on route path) receives this CP1, it appends its own CP to CP1, and forwards it. In this manner, we can reduce the number of transmissions drastically as compared to the scenario when nodes were simply flooding the individual CPs. Finally, when the CPs are received at the BS, the BS begins analysing them to determine the ID of the malicious node(s). The mechanism of gathering CPs is described in detail in Section 4.2.6.

**Analysis:** BS analyses CPs by monitoring the number of transmissions sent and received by each node. It then tallies these numbers with the number of data packets received. Finally, using a threshold (= 4/10 packets, which is decided by running the system repeatedly), the BS can identify the malicious nodes on the routing path between itself and the source router. This process is explained in detail in Section 4.2.7.

**New path:** the BS then sends the information about malicious nodes into the network using the trickle protocol of contiki (*reliable single-source flooding*). Upon receiving this message, every node in the network drops any packet (RREQ/RREP or data) ‘originating from’ or ‘forwarded by’ malicious node(s). Thus, the new path request (RREQ) by the

---

![Figure 4.11: Selective forwarding: (a) detection and correction process, (b) New Path formation](image)
BS results in a routing path that is free from any malicious node(s) as shown in figure 4.11b.

In the following sections details of control packet collection and packet analysis at the BS are presented.

### 4.2.6 Packet Gathering

Packet gathering refers to the collection of required data (patient information) at the BS. The mechanism used for gathering data may be flawed with inefficient energy consumption and unbalanced load distribution. For e.g., energy consumption may be increased if a node transmits data directly to the BS thereby avoiding its next-hop malicious node. Therefore, in this section a local solution for a scenario where control packets (CPs) may be dropped during data gathering stage is discussed. Here ‘local’ refers to radio transmissions without long range capability of reaching the BS.

Figure 4.12 illustrates the mechanism required to solve the above-mentioned problem. The workings of the mechanism are distributed to two layers, the application layer and the RDC layer. The application layer re-organises the network if a node (say node $i$) is dead or drops packets. This layer uses a timeout interval and invokes the RDC layer to monitor the node’s neighbour. If, within this specified time, a packet is not sent by the node’s neighbour, then the current node $i-1$ floods the network with the same packet in a controlled fashion.

![Figure 4.12: Fix for scenario when malicious node drops CPs](image)

This flooding does not include the nodes that are the previous hop of the current node, thereby, reducing traffic to some extent. When the node $i+1$ receives the netflood, it cancels the rebroadcast and forwards the desired packet onto its next hop. Thus, resuming the network mesh protocol and avoiding a DoS attack.

Looking at previous work related to gathering mechanisms, authors of [102] present a mechanism called AntChain. In this mechanism, every node aggregates data from its previous hop, attaches its own data and forwards it onto the next hop. If a node does not receive data then in the next round it will assume that the neighbour is dead, and transmit data directly to the BS. This implies that more energy will be consumed due to a long-
range single-hop transmission. However, in the presented method, the adjustment is purely local. The system sends a local broadcast \textit{(netflood)} instead of a longer range unicast to the BS, thus, the network will remain multi-hop, and the energy consumption will be lower. This is supported by the fact that long range single hop communication consumes more energy than multi-hop [103]. Following this process the system reaches the next stage of the protocol i.e. packet analysis at the BS.

4.2.7 BS Analysis

During the detection phase, it is the role of the base station to analyse the received CPs and identify which node is misbehaving. This is done through a two-pass analysis over an array containing the CPs. The analysis of CPs received from the suspected malicious nodes are skipped (marked by ‘x’, Table 4.1 and Table 4.2). This is done to decrease the false positives and the false negatives. The values of nodes marked with ‘-’ indicate no decision either due to presence of a neighbour malicious node or due to the position of that node in the network being \textit{first or last}. The resulting ‘flags-array’ with bit set to zero indicates the malicious node(s).

Figure 4.13 shows the first pass, which consists of the BS verifying the neighbour DATA packets monitored by each node. Following is an example for the analysis of CPs:

1) \textbf{Threshold} is set to the expected number of packets at the BS, for a timeframe
2) \textbf{Since} the BS and source (S) \textbf{are trusted} (Section 4.2.5), the flags-array bit for these nodes are set to 1 (see row P1, Table 4.1).
3) \textbf{If} the DATA count, monitored by node \( j \) (where \( j \in [S, S+3] \)), is within the threshold range \( (i.e., \text{DATA count} - \text{Threshold} < \text{Expected no. of packet drops}) \)
   \hspace{1cm} \textbf{THEN} set flag for node \( j+1 \) equal to 1, \textbf{ELSE} equal to 0.

![Figure 4.13: Test-Bed Layout. The arrows represent DATA packets, and the bolts represent monitoring the neighbour node’s DATA sending habit.](image)

Table 4.1 shows a sample first pass for the case when node S+3 is malicious. Here P1 refers to Pass 1 and PM refers to packet monitoring of the neighbour node i.e. number of data packets forwarded by the neighbour node i.e. j+1.
This process will set all or some flags, depending upon single or collaborative SF, for nodes S+1 to S+4. Furthermore, the second pass is used to verify and/or fill the flags missed by the first pass.

In the second pass, every node that has its control packets being analysed is not being judged simultaneously. This node may be judged only by the CPs of other nodes. The pass goes from BS side to source side as follows:

1) **Threshold** is set to the number of packets received at the BS (say 4, see row PM of BS in Table 4.1).

2) **If** the ACK count monitored (AM) by node S+4 is within the threshold range, **Table 4.2, THEN** Set flag for S+3 to 1.

3) **Else if** the same ACK count is not within the threshold range **and**
   a. **If** the DATA count monitored by node S+2, **Table 4.1**, is out of range with respect to the no. of ACK packets monitored by BS, **Table 4.2, i.e. node S+2 is lying about no. of packets forwarded by S+3, then**
      i. Set flags for S+2 and S+3 to 0
      ii. Adjust the Threshold
   b. **Else if** the ACK count monitored by S+4, is **not** within the range of the DATA count monitored by node S+2 **then**
      i. Set flag for S+3 to 0.
      ii. Adjust the Threshold

Finally, the malicious node(s) can now be identified by the corresponding zeroes in the flags bit array, **Table 4.2**.
4.3 Details on Protocol Modifications

The functioning of the SF countermeasures in Section 4.2.5 is facilitated by modifications to the underlying protocols of Contiki OS. The lowest layer i.e., the radio duty cycling layer (RDC), which is on top of the data link layer (DLL) [104] has been modified to incorporate the neighbour monitoring process of the presented system.

In the RDC layer, the neighbour addresses are added to the 802.15.4 ACK as shown previously in figure 4.9(b) in Section 4.2.5. This eases the job of nodes in capturing the correct ACK packet. In addition, the ACK capture and DATA capture functions are added to the code. These are responsible for managing the captured ACK and DATA packets by keeping separate associated counters. In this fashion a node has the knowledge of the number of packets sent and/or received by its neighbour(s).

Moving on to the mid-layers. The route-discovery is responsible for setting up a dynamic route between source and destination. This layer is modified to reverse the roles of RREQ and RREP packets, and to include the hashing scheme (Section 4.2.3). The hash, in this case, is calculated by upper layers and verified by this layer. If this verification results in a positive outcome, a RREP packet is send back to the BS. Thus, forming a foundation for the data communications. If not, then the connection is dropped. Additionally, this layer is responsible for tallying the received route packets against the malicious node list. If a RREQ or RREP packet is received from one of these malicious nodes, the layer drops the packet and awaits a packet from one of the redundant mesh nodes.

Next is the mesh protocol layer. This layer provides its own forwarding mechanism built on top of the route-discovery layer. It includes methods to send periodic data, forward data and receive data. This layer monitors route changes and/or breaks in route, and if detected then route discovery is re-initiated (from the BS).

4.4 Results

To better understand the protocol, its functionality using different metrics are presented.

The first metric is network latency (Section 4.4.1), which shows the time difference between detection of SF, and receiving of ‘malicious node information’ at source router. This metric is described for both Single SF and Collaborative SF attacks. The second metric analysed is the accuracy of the SF defence mechanism (Section 4.4.2). Here the metric is examined with the help of both Tmote-sky simulation (Cooja) and openmote real-time analysis and implementation. The third metric is Power/Energy consumption and current consumption. The former is derived for the secure routing protocol running
on Contiki Cooja’s emulation of Tmote-Sky (Section 4.4.3), and the latter is derived for the secure routing protocol running on openmote nodes with Contiki OS in real-time (Section 4.4.4) i.e. the nodes are active and positioned in the university’s hallway. This is done to simulate a clinical/home patient monitoring system environment.

The system considers interference from other devices or technologies while analysing the above-mentioned metrics. In the case where Tmote-Sky simulation (Cooja) is used, the transmission and interference range are set to 50m and 100m respectively. While the real-time test-bed deployment of openmote has variable transmission and interference range. This is because of factors like indoor environment, temperature variations, node placement, antenna, other ISM devices and people passing by the nodes. However, the programmable output power was configured to range between -24dBm to 0dBm, which means the maximum transmission range, in this case, is 1 meter.

### 4.4.1 Latency: Network

In this section latency in the routing path is discussed. This latency is calculated by subtracting the time at which a node receives the detection information, and the time at which the same node receives the ID of the malicious node from BS. This latency is illustrated for both single and collaborative selective forwarding attack, Table 4.3 and Table 4.4 respectively. The columns are described as follows:

1) ‘**No of nodes**’ column shows malicious nodes (marked with *), data flow using arrows, and nodes on the routing path using ‘PN’ while redundant nodes for future new paths using ‘RN’, figure 4.16.

2) ‘**Detection at**’ (D): column shows the time (ms) at which SF detection information reaches a node,

3) ‘**Mal info received**’ (M): shows the time (ms) at which the malicious node list reaches the same node,

4) ‘**Latency**’: column calculates the latency by subtracting (D) from (M).

The following observations were noted:

1) From the tables, it is seen that nodes closer to the BS update information quicker compared to nodes farther away.

2) The maximum latency for Single SF detection and correction was 4767ms for the worst-case scenario.

3) When the malicious node drops the control packets, the latency is higher (19650ms). This is because the intermediate nodes must use netflood in-order to deliver the control packets to their destination (Section 4.2.6)
4) Finally, malicious nodes are detected maximum two at a time for Collaborative SF

Table 4.3: Single Selective Forwarding

<table>
<thead>
<tr>
<th>Type</th>
<th>No. of nodes</th>
<th>Node ID</th>
<th>Detection at (ms) (D)</th>
<th>Mal info received (ms) (M)</th>
<th>Latency (ms) (M - D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single SF</td>
<td>5PN + 2RN</td>
<td>4</td>
<td>722223</td>
<td>725201</td>
<td>2978</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>722268</td>
<td>724013</td>
<td>1745</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>722311</td>
<td>723345</td>
<td>1034</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>722399</td>
<td>722506</td>
<td>107</td>
</tr>
<tr>
<td>Single SF</td>
<td>6PN + 1RN</td>
<td>4</td>
<td>742919</td>
<td>747686</td>
<td>4767</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>742970</td>
<td>746498</td>
<td>3528</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>743021</td>
<td>745795</td>
<td>2774</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>743071</td>
<td>745127</td>
<td>2056</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>743173</td>
<td>743295</td>
<td>122</td>
</tr>
<tr>
<td>Single SF**</td>
<td>5PN + 1RN</td>
<td>4</td>
<td>719981</td>
<td>739631</td>
<td>19650</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>720026</td>
<td>738591</td>
<td>18565</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>720069</td>
<td>737921</td>
<td>17852</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>736383 (with netflood)</td>
<td>736482</td>
<td>99</td>
</tr>
</tbody>
</table>

*Malicious Node, **Malicious node drops control packets

Table 4.4: Collaborative Selective Forwarding

<table>
<thead>
<tr>
<th>Type</th>
<th>No. of nodes</th>
<th>Node ID</th>
<th>Detection at (ms) (D)</th>
<th>Mal info received (ms) (M)</th>
<th>Latency (ms) (M - D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaborative SF</td>
<td>5PN + 2RN</td>
<td>4</td>
<td>734278</td>
<td>737233</td>
<td>2955</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>734323</td>
<td>737057</td>
<td>2734</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>734454</td>
<td>734561</td>
<td>107</td>
</tr>
<tr>
<td>Collaborative SF Round 1</td>
<td>5PN + 4RN</td>
<td>4</td>
<td>698901</td>
<td>703183</td>
<td>4282</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>699077</td>
<td>699184</td>
<td>107</td>
</tr>
<tr>
<td>Collaborative SF Round 2</td>
<td>5PN + 4RN</td>
<td>4</td>
<td>1710799</td>
<td>1712192</td>
<td>1393</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>1710931</td>
<td>1711184</td>
<td>253</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1710975</td>
<td>1711082</td>
<td>107</td>
</tr>
</tbody>
</table>

*Malicious Node

4.4.2 Accuracy

In this section the accuracy of the selective forwarding defence mechanism is discussed. It is classified into two categories i.e. detection and correction. The former refers to the
ability of the system to accurately detect the malicious node(s), and the latter refers to the ability of the system in forming a malicious-node free path.

The accuracy is calculated by examining the network layouts in Table 4.5, 4.6, and 4.7.

<table>
<thead>
<tr>
<th>Original Path</th>
<th>New Path</th>
<th>Detection</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tmote-sky (Emulated on Cooja)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4,2,3*,1</td>
<td>4,2,5,1</td>
<td>Yes</td>
<td>1 attempt</td>
</tr>
<tr>
<td>4,3*,7,6,2,5,1</td>
<td>4,8,7,6,2,5,1</td>
<td>Yes</td>
<td>1 attempt</td>
</tr>
<tr>
<td>4,8,7,3*,5,6,2,9,1</td>
<td>4,8,7,10,5,6,2,9,1</td>
<td>Yes</td>
<td>2 attempts</td>
</tr>
<tr>
<td>4,6,3*,8,7,5,10,2,9,1</td>
<td>4,6,12,8,7,11,10,2,9,1</td>
<td>Yes &amp; False Positive: ID 5</td>
<td>1 attempt</td>
</tr>
<tr>
<td>4,10,9,5,7,8,6,2,3*,1</td>
<td>4,10,9,5,7,8,6,2,11,1</td>
<td>Yes</td>
<td>2 attempts</td>
</tr>
</tbody>
</table>

Openmote (Hardware)

<table>
<thead>
<tr>
<th>Original Path</th>
<th>New Path</th>
<th>Detection</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>4,3*,6,1</td>
<td>4,2,6,1</td>
<td>Yes</td>
<td>1 attempt</td>
</tr>
<tr>
<td>4,6,3*,1</td>
<td>X</td>
<td>No: False Negative</td>
<td>N/A</td>
</tr>
<tr>
<td>4,3*,2,6,1</td>
<td>X</td>
<td>Yes</td>
<td>N/A - Due to limitations on no of nodes</td>
</tr>
</tbody>
</table>

A* Represents the malicious node, A Represents the redundant node

<table>
<thead>
<tr>
<th>Original Path</th>
<th>New Path</th>
<th>Detection</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tmote-Sky (Emulated on Cooja)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 4,2,3*,7,5,1 | 4,2,8,7,5,1 | Yes | 1 attempt:
| 4,2,3*,7,5,8,9,10,1 | 4,2,11,7,5,8,9,10,1 | Yes | 1 attempt:
| 4,2,7,5,3*,9,10,1 | X | Yes | N/A |
| 4,2,5,9,8,3*,10,1 | 4,2,5,8,8,7,10,1 | Yes | 3 attempts: |
| 4,2,5,9,8,7,3*,10,11,1 | 4,2,5,9,8,7,12,10,11,1 | Yes | 1 attempt |
| 4,2,5,9,8,7,3*,12,10,11,1 | 4,2,5,9,8,7,13,12,10,11,1 | Yes | 4 attempts |
| 4,2,5,7,8,9,3*,12,10,11,13,1 | 4,2,5,7,8,9,14,12,10,11,13,1 | Yes | 1 attempt |

A* Represents the malicious node, A Represents the redundant node

In the tables, the first column shows the original data path; numbers marked with an asterisk refer to the malicious node ID(s). The second column shows the new path formed free of any malicious node ID(s) as the redundant node (RN) takes the place of the malicious node, this is shown in bold formatting. The third column notifies whether detection was successful or not and reports false alarms. Finally, the fourth column shows the number of attempts it took to form the new path in second column.

The implementation set-up is as follows: a certain node $i$ is configured to drop packets selectively. It may or may not lie about the control packet (CP) information it
generates. Additionally, for the collaborative case, node $i-1$ is configured to lie about the number of data packets send by node $i$. This is done to cover up the malicious intent of node $i$.

Table 4.7: Accuracy check for collaborative SF - nullRdc

<table>
<thead>
<tr>
<th>Original Path</th>
<th>New Path</th>
<th>Detection</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tmote-sky (Emulated on Cooja)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4, 6, 5*, 3*, 2, 1</td>
<td>4, 6, 8, 7, 2, 1</td>
<td>Yes</td>
<td>3 attempts</td>
</tr>
<tr>
<td>4,6,5,7*,3*,2,1</td>
<td>4,6,5,9,8,2,1</td>
<td>Yes</td>
<td>1 attempt</td>
</tr>
<tr>
<td>4,2,7,5,8*,3*,6,1</td>
<td>4,2,7,5,10,9,6,1</td>
<td>Yes</td>
<td>3 attempts</td>
</tr>
<tr>
<td>4,6,9*,5*,7,3,2,8,1</td>
<td>4,6,11,5*,7,3,2,8,1</td>
<td>Yes</td>
<td>1 attempt, 1 Error</td>
</tr>
<tr>
<td>4,6,11,9*,5*,2,8,7,3,1</td>
<td>4,6,11,10,12,2,8,7,3,1</td>
<td>Yes</td>
<td>2 attempts</td>
</tr>
<tr>
<td>4,7,11,9*,3*,2,8,6,5,1</td>
<td>4,7,11,12,13,2,8,6,5,1</td>
<td>Yes</td>
<td>3 attempts</td>
</tr>
<tr>
<td>4,12,11,9*,3*,1</td>
<td>4,12,11,14,3*,1</td>
<td>Yes</td>
<td>1 attempt, 1 Error</td>
</tr>
<tr>
<td>4,12,9*,3*,11,1</td>
<td>4,12,14,13,11,1</td>
<td>Yes</td>
<td>3 attempts</td>
</tr>
<tr>
<td>4,5,6*,2*,3,8,11,1</td>
<td>4,5,13,10,3,8,11,1</td>
<td>Yes</td>
<td>2 attempts</td>
</tr>
<tr>
<td>4,9,6*,2*,10,8,1</td>
<td>4,9,11,3,10,8,1</td>
<td>Yes</td>
<td>5 attempts</td>
</tr>
<tr>
<td>4,2,7,9*,5*,3,6,1</td>
<td>4,2,7,11,10,3,6,1</td>
<td>Yes</td>
<td>3 attempts</td>
</tr>
<tr>
<td>I) 4,2,10,9*,5*,3,6,1</td>
<td>I)4,2,10,7*,8,3,6,1</td>
<td>I)Yes: 5 &amp; 9</td>
<td>1 attempt</td>
</tr>
<tr>
<td>II) 4,2,10,7*,8,3,6,1</td>
<td>II)4,2,10,11,8,3,6,1</td>
<td>II)Yes: 7</td>
<td>2 attempts</td>
</tr>
<tr>
<td>4,6*,2*,8,1</td>
<td>4,9,10,8,1</td>
<td>Yes</td>
<td>2 attempts</td>
</tr>
<tr>
<td>Openmote (Hardware)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4,6*,2*,1</td>
<td>4,8,10,1</td>
<td>Yes</td>
<td>1 attempt</td>
</tr>
</tbody>
</table>

* Represents the malicious node, * Represents the redundant node
1 Error: node not identified as malicious; 2 Infiltration during the detection process

Following this set-up, the protocol was analysed for 29 different scenarios out of which, 4 scenarios were implemented on the openmote hardware/test bed due to node availability and location limitations. The rest were implemented on the cooja simulator for the Tmote-sky platform. The scenarios include single SF, collaborative SF, different number of nodes, and different malicious node IDs.

As shown, the protocol detects a selective forwarding attack with 96.5% accuracy, and forms a malicious-node free path with 83% accuracy. In addition, the protocol had 0.03% of false positives and 0.06% of false negatives only.

This increases confidence in the future use of this protocol in medical systems as it has a high rate of accuracy, low false alarm rate, and does not consume much extra power/energy to facilitate the inclusion of security. The protocol may also be used in other WSN applications or scenarios that are susceptible to SF and Black Hole attacks.
4.4.3 Power consumption: Tmote Sky (Cooja)

The first scenario comprises of a network consisting of Tmote sky (MSP430+CC2420) nodes on the Contiki Cooja simulator. This set-up builds around a unit disk graph medium (UDGM): i.e. Distance loss radio propagation model. This model includes interference with the capability of transmitting (TX) and receiving (RX) the packets with a success ratio probability [105].

Here, the power consumption of the modified routing protocol is determined using the powertrace-profiler [106] to better understand the system, its lifetime, and its availability. Then the energy consumption is derived from these graphs and compared against the system in [90]. This is done because the energy efficiency of routing protocols affects the system lifetime and system availability.

The power consumed by the source node during two phases of the protocol i.e. normal phase and detection phase is provided. Where the former refers to the operation of the routing protocol without any interference from malicious node(s), and the latter refers to the following:

- process of malicious node detection at BS,
- propagation of this information to the routers (Access points),
- gathering and analysis of CPs at the BS, and
- propagation of the generated malicious-node list

These phases have different power consumption as shown in figure 4.14 and figure 4.15. The former illustrates the average power during the normal phase of the protocol i.e. 62.780mW, and the latter illustrates the same for the detection phase of the protocol i.e. 62.932mW. As it can be seen there is a difference of only 152μW. So, it is inferred that the power consumption during the detection phase is almost the same as the power consumption during the normal phase. This is because the protocol analysis has been shifted from the routers to the base station. Thus, relieving the routers from unnecessary processing overhead and subsequent power consumption.
Furthermore, from figure 4.14 and figure 4.15, the time spend by the source node in normal and detection phases, are 135s and 665ms respectively. Thus the average energy consumption (given in Joules as this is the unit used in [90]) is 8.5J and 41.8mJ for normal and detection phase respectively. Here, the detection phase consumes 8.46J more than the normal phase. Thus the modified routing protocol has an advantage compared to the system in [90] because the latter consumes approximately 200J of energy more in its detection phase than its normal phase.

Comparing this work with the original protocol that is built-on Contiki OS, the latter has an average power consumption of 62.639mW for the source node. This implies that
the modified protocol presented here consumes an excess of only 141μW for normal mode and 293μW for detection mode.

### 4.4.4 Current consumption: Openmote

In this section the second scenario, test-bed implementation, is analysed. This makes use of motes distributed through one of the hallways of our university, like the positions represented in figure 4.16. This is done to emulate the home/clinic network environment.

This network consists of two types of connections i.e. wired and wireless. Wired connection between the source node and laptop is provided through an OpenBase board [107]. This board connects the motes to the laptop using a serial port that is interfaced through a FTDI FT232R chip. The wireless network operates at 2.4 GHz and is compatible with the IEEE 802.15.4 standard.

The basic process is that the source node (ID: 4) is connected to a laptop, for monitoring power consumption, and the other nodes follow the order as shown in figure 4.16. The source node sends patient data periodically towards the BS and the access point’s act as routers. Access point 3 is an observer i.e. it does not take part in the data sending process but acts as a redundant node for the detection phase of the routing protocol. At the same time, the whole network is being monitored on an individual level by each node with timing intervals that check whether the neighbours forward or drop any packets. At the end of this process any malicious node should be detected (if any), and the route corrected.

Following are the measurements taken for the two phase’s i.e. normal phase and detection phase. Here, the current was measured using the Agilent 66321D mobile communications Direct Current (DC) source [108] and the 14565B device characterization software [109], with a sampling rate of 1364Hz. figure 4.17 shows the current consumption for the detection phase on an openmote.
Measurement begins as soon as the Light Emitting Diode (LED) lights up, and finishes just before the LED turns off i.e. end of detection phase. The measurement shows the average current to be 21.0579mA.

Figure 4.18 illustrates the current consumption for the normal phase-on of the openmote. Here, the average current consumed by the source node (inclusive of all processes / components), is 20.223mA. From figure 4.17 and figure 4.18 there is an overhead of 834.9μA during the detection phase of the protocol.

Considering this overhead, the worst-case scenario calculations can be made. This scenario refers to the detection phase being invoked after every ten data packets received at the base station. In this case, a battery of 1100mAh will last approximately 49hrs; considering an excess of 10% to avoid the battery from going into deep discharge.
In addition to the current consumption, the memory footprint is shown in figure 4.19. Here the original protocol consumes 32,815 bytes i.e. 6471 bytes of RAM, and 26344 bytes of ROM on the openmote platform. The modified routing protocol consumes a total of 40,322 bytes i.e. 7718 bytes of RAM, and 32604 bytes of ROM on the openmote. This amounts to an excess of 7,507 bytes, for the modified protocol, when compared to the original protocol.

The advantage provided by the modifications proposed here is that extra security is provided with low overhead in regards to energy/power and current consumption. However, the disadvantage is that the addition of security takes up to 7.5kB more memory than the original protocol.

![Figure 4.19: Memory footprint for the program on an openmote](image)

### 4.5 Security Implemented

The inclusion of security is a vital parameter in the construction of a medical WSN. It is important to guarantee patient’s safety in addition to safeguarding the privacy of their data. Here the security of the secure routing phase of the presented system is outlined briefly. This is an important step in the construction of the system as it helps in identifying any potential weaknesses in the systems security:

**Confidentiality:** is provided by AES GCM 256 cipher, and is brought over from cluster elections phase (Chapter 6). Hence, the data handled by the routing protocol is already encrypted at the source i.e. sensor nodes represented on a patient’s body.

**Integrity:** is provided for both control and data packets of the routing protocol by making use of the hash based approach as described Section 4.2.3. Where the BH and SF attacks are launched, and countermeasures are provided.
Availability: of service is ensured as the system is fortified against route-breaks, route-changes, replay attacks, and Denial of Service attacks as described in Section 4.2.

4.6 Summary

The security of data routing has been an unnecessary overhead in most of the published works in the field of WSN. As mentioned before, the DoS attack on data routing can have a significant impact on a medical based WSN systems. In this chapter an approach was presented to provide schemes that prevent black hole attacks, and detect and correct selective forwarding attacks for the system presented in this thesis.

The defence mechanism used against Selective forwarding attack provides a suitable way to deal with both single and collaborative attacks. This is useful because real world attacks are often collaborative by nature between malicious nodes.

The detection rate of the proposed SF prevention system was found to be over 96%, which is an improvement over the accuracy of the system presented in [90]. While the correction rate was found to be 83%, as mentioned in Section 4.4.2, the system examined took into consideration radio interference, which is an added advantage for real-world implementations. This is because the system presented in this thesis was also implemented in a real-time test-bed deployment. This deployment helps to demonstrate the dependability and reliability of the presented system as simulations alone cannot capture the dynamics and complexity of the real-world.

This part of the presented system is published in [44], [110]. It provides several features that can be useful in the real-world implementations of WSN in healthcare industries. For e.g., low energy consumption, high accuracy, confidentiality, integrity, and availability. These are important as they facilitate risk assessment and management of the system in addition to providing data and routing security.

Finally, this phase forms the fundamental basis for working towards a full WSN system for patient monitoring with key management as shown in the following chapter.
Chapter 5

System Key Management

Previous chapter discussed the main security measures associated with the presented system i.e. secure routing. Following this, the next phase of the presented system incorporates key management between routers and cluster, and between PC and router. Additionally, the BS connects to the PC via a broker on the Internet. This gives rise to a full-fledged patient monitoring system as shown in figure 5.1, capable of being controlled from any part of the world\(^7\).

![Figure 5.1: End-to-End Secure WSN system: Communications standards used and security implemented](image)

A patient monitoring system [44], [111] may require higher security measures as compared to other applications using the WSN as shown in Chapter 3 because a patient monitoring system is susceptible to confidentiality breach, denial of service and unauthorized access. To prevent these attacks this chapter works towards a complete secure patient monitoring system that is secure for different components and aspects of the network. This chapter provides the key management features incorporated into the system in conjunction with Confidentiality, Integrity and Availability (CIA triad) [112].

\(^7\) If there is internet connectivity and the PC application runs on a Linux environment
This system is used for patient monitoring and is based on an end-to-end secure WSN model. Here, several communication standards are used for the different devices. The standards are Message Queuing Telemetry Transport (MQTT) [79], universal asynchronous receiver/transmitter (UART) and RIME [49]. Each of these have already been described in Section 3.1.2.

The devices associated with these standards, and used in the presented system are:

1) **Subscribed PC**: a Linux PC subscribed to a particular MQTT topic through a broker (*mosquitto*) on the cloud network. This PC runs the Linux operating system Ubuntu 14.04 LTS and a custom python program to communicate with the broker.

2) **Intel Edison Gateway** [113]: This device runs the Debian (Ubilinux) OS and a python program to communicate with the cloud and BS.

3) **Openmote** [45], [107]: This device runs the Contiki OS [47] and is used for the WSN side of the network.

The reason for combining the WSN side of the system with internet connectivity is that: (1) it allows for data to be send anywhere across the world. (2) The hospital staff / doctor can take immediate action if required. For example, sending an ambulance or administering a drug dosage. (3) The sensors and routers can be managed remotely without manual interference at the physical locations of the nodes, and (4) Increase in automation of the system means that the hospital staff can be free for more emergency cases, if required.

The rest of the chapter is structured as follows: in Section 5.1, current solutions related to the different aspects of WSN systems and key management are discussed. Each solution is explained and then compared against each other in tabular form. In Section 5.2, the prerequisites for system key management are given, this includes devices used, communication standards employed, and security technologies integrated. In Section 5.3, several key management techniques like HMAC-based Key Derivation Function (HKDF) and Elliptic Curve Diffie-Hellman (ECDH) are defined and used in the proposed system. This section also provides results for a real-world implementation of the system on hardware platforms (openmote, Intel Edison, and Linux PC). In Section 5.4, the approaches for seeding a pseudo random number generator (PRNG) are discussed. This is an important phase for the generation of keys that may be used in the key management phase from Section 5.3. Finally, Section 5.5 summarises the whole system with focus on security and the advantages it offers.
5.1 Current Solutions

Table 5.1 shows the current solutions that are discussed in this section.

Table 5.1: Related Work Comparison

<table>
<thead>
<tr>
<th>Related Work Ref.</th>
<th>Scheme Used</th>
<th>Frequency (MHz)</th>
<th>Time (ms)</th>
<th>Energy (mJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ECDH &amp; RSA Comparison</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presented System</td>
<td>ECDH</td>
<td>16</td>
<td>682.6</td>
<td>21.49</td>
</tr>
<tr>
<td>[54]</td>
<td>ECDH</td>
<td>7.37</td>
<td>--</td>
<td>42</td>
</tr>
<tr>
<td>[133]</td>
<td>ECDH</td>
<td>13</td>
<td>2718.35</td>
<td>80.05</td>
</tr>
<tr>
<td>[18], [135]</td>
<td>RSA-1024*</td>
<td>14.7456</td>
<td>26.494</td>
<td>1.81</td>
</tr>
<tr>
<td><strong>AES Comparison</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presented System</td>
<td>AES-256-GCM</td>
<td>16</td>
<td>0.132</td>
<td>2.252</td>
</tr>
<tr>
<td>[18], [135]</td>
<td>AES-128-CBC*</td>
<td>14.7456</td>
<td>10.44</td>
<td>0.6</td>
</tr>
<tr>
<td><strong>Others</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presented System</td>
<td>HKDF</td>
<td>16</td>
<td>1.2294</td>
<td>0.049</td>
</tr>
<tr>
<td>[118]</td>
<td>S3K (DTLS based)</td>
<td>32</td>
<td>511.65 to 711.11</td>
<td>0.2 to 1</td>
</tr>
</tbody>
</table>

* Calculated using $E = V \times I \times t$ equation, where $V = 3V$, and $I$ was retrieved from datasheet.

2 Calculated using $E = V \times I \times t$ equation, where $V = 3.28V$ and $I$ was measured using Agilent 66321D device [108].

* Datasheet values

In this table, ‘Presented System’ refers to the system presented in figure 5.1 i.e. an end-to-end secure WSN system. The security technologies incorporated in this system are ECDH and HKDF algorithms, and AES Galois Counter Mode (AES-GCM 256) cipher. The background description of these technologies is outlined in Section 3.1.1, while their implementation in the proposed system is described in Section 5.3. In Section 5.3 the experiments corresponding to the security technologies were carried out, and the results thus obtained from that section are combined and integrated into Table 5.1 under the ‘Presented System’ heading. This is done for easier comparison with the other solutions provided in this section.

Changing focus back to the available current solutions: there have been several different works that review WSN systems and/or parts of systems. These systems/components may be attacked in various ways, so there has been considerable amount of research into these systems and their defence mechanisms. For example, the authors in [114] formulate a secure data transmission protocol, and discuss issues relating
to integration of security in medical WSNs. In [115] the authors provide a lightweight and secure system for medical monitoring. The implementation makes use of limited resources hardware, and secures the data transmission through the system while providing authentication via access control mechanisms. Interested readers may refer to other security-related healthcare WSN applications in [13], [111], [116].

Several authors have tried to solve the security challenges that affect WSN and/or IoT systems/components. Authors in [117] provide an overview on the challenges of IoT applications in smart grids related to security and privacy. In [118] the authors present a key management architecture for IoT with constrained resources. They implement an IoT system with a client, a trusted authority, and a resource server to evaluate a key derived from their key management scheme. They have made use of the Contiki operating system, and the CC2538 development kit [119]. Their findings show that a complete handshake duration required to derive a key ranged between 511.65ms and 711.11ms. This is large compared to the key derivation schemes used in our system (HKDF and ECDH) that take 1.22ms and 682.6ms respectively for the generation of a new key. Additionally, the energy consumed by the processes in key management architecture in [118] range from 200 to 1000 μJ. In contrast, the key derivation processes in our system (HKDF and ECDH) consume 49μJ and 21.49mJ respectively.

In [18], the authors compare and contrast existing security schemes for WSN, and introduce a new hybrid scheme. This scheme makes use of multiple key management concepts like trusted third party (TTP), node joining / revocation, and key storage. In contrast, the scheme presented in this paper does not require a TTP as authentication is provided through a hash-based mechanism discussed in Section 4.2.3, and published in [44]. Moreover, this scheme uses RSA to encrypt and decrypt a newly generated key, which can be expensive in the application layer, (up to 2 seconds). Hence, our scheme makes use of ECDH as it takes only 682.6ms to generate new keys.

In [54], the authors introduce an optimised implementation of the National Institute of Standards and Technology (NIST) P-192 Elliptic Curve running on an ATMega 128 Processor at 7.37MHz. The ECDH scheme implemented in this system consumes 12.08 x 10^6 to 12.86 x 10^6 clock cycles, and 42mJ of energy. In contrast, our system uses the NIST P-256 Elliptic Curve for the ECDH scheme (based on the ECC library of contiki [120]), and runs on the CC2538 SoC [121] with a clock frequency of 16MHz. This results in a consumption of 10.92 x 10^6 clock cycles, and 21.49mJ of energy as shown in Table
5.1. Thereby showing an improvement of $1.16 \times 1.94 \times 10^6$ clock cycles and 20.51mJ in energy consumption.

Authors in [122] propose new seeding techniques for a strong foundation when applying security to a wireless sensor system. They provide details on leveraging sensor data for providing secure seeds for a random number generator (RNG), which includes the Blum-Blum-Shub scheme, Washed + Rinsed scheme, Hashing algorithm and raw accelerometer data. In the proposed system, a real-time clock and accelerometer readings are used. These were timed and it was found that the latter is 80 times faster than the former mechanism.

Authors in [123] use a key derivation function for generating keys in an LTE network with the help of SHA-256 and HMAC. The paper lacks any timing or energy consumption measurements. Authors in [124] explain the security associated with HKDF and provide extensive comparisons with other key derivation functions.

### 5.2 Prerequisites for System Key Management

#### 5.2.1 Platform: Openmote

In the system presented here the openmote platform [45] is used. The main reason for using this platform is that it has sufficient RAM and ROM: 32kB and 512kB, and has hardware accelerators available for SHA-256, ECC and AES. This is useful as it speeds up the time required for computations for each of these security schemes. Another reason is that this platform supports multiple Operating Systems that includes the Contiki OS [47], which is the preferred OS for the implementation of the presented system as justified in Section 2.1.5.

#### 5.2.2 Platform: Intel Edison

In addition, the Intel Edison Arduino Breakout Board is utilised. The board is useful as it consists of 1 GB RAM, and a USB Type-A connector. The former is useful to facilitate simultaneous MQTT and UART connections, and the latter is useful for connecting the Openmote BaseStation directly to the Edison board. Additionally, Edison uses the Debian ubilinux instead of the default Yocto Linux. This is done to provide more freedom during programming, and for certain features like the paho MQTT client for python, which were unavailable on Yocto Linux during the implementation of the presented system.

#### 5.2.3 Platform: PC

A PC application is used for receiving the encrypted data and decrypting it using AES-GCM 256, which is based on modifications to the library provided by the author in
This application connects to a PC on the Internet running a Mosquitto Broker \cite{80} via the MQTT protocol (Section 3.1.2) with TLS (Section 3.1.1). A screenshot of this application is presented in Chapter 7.

5.2.4 Platform: Mosquitto Broker (via Internet)

The services of the Mosquitto broker are incorporated into the presented system on a different PC than the Subscribed PC. The MQTT running over this server PC (broker) makes use of the encrypted port 8883 in conjunction with TLS. The corresponding X509 certificates for the server (Broker), and client (PC and Gateway) are signed by the root CA, which uses a self-signed certificate for testing purposes. This is done because it would not be possible to sign the server and client certificates as it’s not possible to obtain a private key of a public server\textsuperscript{8}.

5.2.5 System Work-flow

The system model used defines an end-to-end system that can connect WSN enabled sensors on a patient to any PC in the world while avoiding direct access to the sensors from the internet. This technique is advantageous because it mitigates against unauthorized IP-access of the sensors or routers from the internet, which may lead to various security attacks \cite{83}, \cite{126} while keeping the sensors and routers connected to the internet.

As mentioned previously, the architecture comprises of three communication standards i.e. MQTT, UART and RIME as illustrated in figure 5.1. MQTT provides a lightweight publish/subscribe model for communication between the Gateway and a Subscribed PC (PC subscribed to a MQTT Topic). It also has a small overhead for message transport, provides Quality of Service (QoS), and makes use of TCP/IP for network connectivity \cite{79}. RIME, on the other hand, is a layered communication stack with low complexity of implementation for sensor network protocols, and low memory footprint (100-226 bytes) \cite{49}. This is ideal for the sensors’ side of the system due to their resource constraints.

\textsuperscript{8} Public server running on the following address: www.iot.eclipse.org.
Chapter 5. System Key Management

Figure 5.2 illustrates the work flow of the system, which follows the following steps:

1) The subscribed PC sends a ‘start’ command to the BS in the WSN network.
2) The BS initiates a secure on-demand routing protocol [44], [127] to set up a route with Router R.
3) Following this, ECDH phase I commences (Section 5.3.1).
4) Router R broadcasts a ‘Hello’ packet with its public key (Q_r) to the sensors.
5) Following the cluster head (CH) elections [44], [128], the second phase of the ECDH (Section 5.3.1) commences. This results in the generation of a shared key (K_s) between the router and the sensors.

Now, the sensors commence data sampling as shown in figure 5.3.

6) Each sensor i, encrypts its sampled data D_i using AES-GCM 256bit

   \[ \{D_i\}K_s \parallel T_s \]  

   (5.1)

   Where K_s is the session key, and T_s is the generated authentication tag.
7) Each sensor’s encrypted data is aggregated at the CH, and send to Router R. The aggregation takes place in the following manner:
\[ \{D_1\}K_s \parallel T_s \parallel \{D_2\}K_s \parallel T_s \parallel \ldots \parallel \{D_n\}K_s \parallel T_s \]

represented as \([\{D_1\}K_s \parallel T_s]_n\) in figure 5.3. \hspace{1cm} (5.2)

Here, \(n\) is the number of sensors in the cluster

8) Following this, Router R decrypts the data and verifies its authentication tag.

9) R then encrypts this aggregated data with a pre-loaded key \(K_m\), and sends it to the BS.

\[ \{D_1 \parallel D_2 \parallel \ldots \parallel D_n\}K_m \parallel T_m \] \hspace{1cm} (5.3)

10) The BS, forwards this to the PC via the gateway.

11) Finally, the PC verifies tag \(T_m\), and decrypts the encrypted data using AES-GCM with the pre-loaded key \(K_m\).

As shown, data / commands travel from different entities (sensors, routers, Gateway, Cloud, and PC), and through different communication channels (RIME, Internet, and UART). This raises security concerns that call for extensive measures in order to safeguard the system, the data, and the people involved. Again, from figure 5.1, the different security measures implemented in our system are:

1) Confidentiality: AES-GCM 256 bit [57].

2) Data Integrity: Using AES-GCM’s one-pass authentication.

3) Entity Authentication between the BS and the Router R using the HASH-based scheme presented in Section 4.2.3 [44]. Authentication between PC and the Gateway G using pre-loaded keys.

4) Key Management using ECDH [53] and HKDF [55] between the Router R and the sensors, and between the PC and Router R, respectively.

These measures, and their usage in our system have already been detailed in Section 3.2.1.

\section*{5.3 Key management}

All the above processes are fruitless if the keys are not updated securely. This requires the use of key management, which is implemented in the system in two phases: (1) Router \(R\) to the Cluster of sensors \(S\), and (2) Subscribed PC \(P\) to Router \(R\). Following are the key notations used:

1) \(K_s\): Session key generated between \(R\) and \(S\)

2) \(K_m\): Pre-loaded master key between the \(P\) and \(R\)

3) \(K_{m_{\text{new}}}\): New Key generated between \(P\) and \(R\)

\(K_s\) key is used to encrypt and authenticate data between the router \(R\) and cluster of Sensors \(S\) using the AES-GCM 256 cipher. This key is always generated using the ECDH algorithm. \(K_m\) key between \(P\) and router \(R\) is pre-loaded for first time use. Following this,
it is always updated using the HKDF mechanism, which results in a new session key $K_{m_{\text{new}}}$ (see Appendix B.4 for important header file).

5.3.1 Key Updates

The key management process is implemented in the system in two phases:

**Router $R$ to the Cluster of sensors $S$**: Here, the key management process encompasses key generation, and key update, with usage of the ECDH algorithm [53]. When compared to its counterparts like RSA [52], Elliptic curve cryptography (ECC) is used to ensure that the key size is small [129], additionally implementations of ECC are resource efficient [130]. In this system the ECC library provided for cc2538 in the Contiki OS [120] is used to run the ECDH. Following is a brief explanation of the ECDH algorithm:

**Phase 1**

Elliptic curve NIST P-256 curve (secp256r1) is selected as the curve, as it is one of the NIST recommended curves [131] in the 256 bit ECC family. This curve is generated from random values, and is implemented in the ECC library provided in Contiki.

<table>
<thead>
<tr>
<th>Table 5.2: ECDH Phase I between Router $R$ and Cluster of Sensors $S$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Router $R$</strong></td>
</tr>
<tr>
<td>Choose Random Secret =&gt; Generate private key $k_r$</td>
</tr>
<tr>
<td>Calculate public key $Q_r = k_r \cdot G$</td>
</tr>
<tr>
<td>Broadcast $Q_r$ to the cluster of sensors $S$</td>
</tr>
</tbody>
</table>

$G$ is the generator point of elliptic curve secp256r1

*Figure 5.4, Point 1, **Figure 5.4, Point 3
*Figure 5.4, Point 2, ***Figure 5.4, Point 4

**Figure 5.4:** ECDH between router and sensors.

**Phase 2**
1) Router $R$ calculates shared secret $(K_s) = k_r.Q_c$ (Figure 5.4, point 5. of Router):

   Here Since $Q_c = k_c.G$, therefore $K_s$ can also be written as $k_c.k_r.G$ (5.4)

2) Sensors $S$ calculate shared secret $(K_s) = k_c.Q_r$ (Figure 5.4, point 5. of Cluster of Sensors):

   Here since $Q_r = k_r.G$, therefore $K_s$ can also be written as $k_c.k_r.G$ (5.5)

Eq. 5.4 and Eq. 5.5 show that the shared secret $(K_s)$ is same for both parties. This is because the scalar multiplication $(.)$ observes the commutative law of mathematics. Thereby making $k_c.k_r.G$ and $k_r.k_c.G$ equal. Therefore, the new session key generated $K_s$ is same on both sides (from figure 5.4, Eq. 5.4 and Eq. 5.5). This session key is used to encrypt data from the sensors to the router using AES-GCM 256 bit, and is periodically updated. This update time is set arbitrarily and can be adjusted to suit the needs of the application.

**Router $R$ to PC side:** This part of the network implements the HKDF process in accordance with the IETF standard RFC 5869 [55], and the HMAC process in accordance with the IETF standard RFC 2104 [132]. In these implementations, the SHA hardware accelerator for cc2538 in the openmote is used, and python programming language in the PC is used.

The reason for using HKDF instead of ECDH is that the latter would have to be implemented across different platforms running different operating systems, and the exchange would have to take place through the cloud using MQTT (see figure 5.1). This would put an excessive workload on the WSN router $R$ that is already handling the ECDH process with the cluster of sensors $S$.

Furthermore, the designated router $R$ updates its key using HKDF, when it receives an ‘update’ command from the subscribed PC via the Gateway (using MQTT protocol) as shown in figure 5.5. The process results in a new key $K_{m\_new}$ generated on both sides (PC and $R$) without sending any information over the network. This saves energy because radio transmissions are reduced as compared to ECDH as shown in the following section. However, the disadvantage is that if the router or the PC is compromised, then the subsequent keys generated may be compromised. So, in-order to reduce this likelihood one may change the ‘opad’, and ‘ipad’ values used in HMAC to secret values instead of the general ones provided in the HMAC RFC 2104 [132]. In this fashion, an attacker may not be able to launch attacks that analyse the cipher text, unless they have knowledge of both the ‘opad’ and the ‘ipad’.
5.3.2 Results

This section presents the results associated with some of the key management techniques mentioned previously, and other cryptographic measures used in our system. These are compared with each other and/or with other systems as mentioned in their respective tables.

Table 5.3 presents analysis of different key management techniques in our presented system, in TinyECC [133] implementation, and in the system presented in [54]. From the table, it can be deduced that the non-optimized ECDH implemented in contiki is more efficient than the one presented in TinyECC without optimizations enabled [133], and also to the one presented in [54].

<table>
<thead>
<tr>
<th>Libraries</th>
<th>( ECDH ) (Openmote 16 MHz)(^1) (Presented System)</th>
<th>( ECDH ) in [133] (imote2 13MHz)(^2)</th>
<th>( ECDH ) in [54] (Atmel 128) @ 7.37 MHz (^3)</th>
<th>HKDF(^4) (Presented System)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Time (ms)</td>
<td>682.6</td>
<td>2718.35</td>
<td>1639 - 1743</td>
<td>1.2294</td>
</tr>
<tr>
<td>Clock Cycles ((\times 10^6))</td>
<td>10.912</td>
<td>35.334</td>
<td>12.08 – 12.85</td>
<td>0.019</td>
</tr>
<tr>
<td>Time for each stage (ms)</td>
<td>N/A</td>
<td>Initialization: 0.04</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Round 1: 343</td>
<td>Key-Establish: 2718.31</td>
<td>Extract: 0.599</td>
<td>Expand: 0.630</td>
<td></td>
</tr>
<tr>
<td>Round 2: 339.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Energy</td>
<td>21.49mJ</td>
<td>80.05mJ</td>
<td>42mJ</td>
<td>0.049mJ</td>
</tr>
<tr>
<td>Notes</td>
<td>ECC hardware accelerator</td>
<td>--</td>
<td>--</td>
<td>Software + SHA hardware accelerator</td>
</tr>
</tbody>
</table>

\(^1\)No optimizations, \(^2\)Optimizations Disabled, \(^3\)NIST P-192 Curve, \(^4\)Custom build
Additionally, from Table 5.3 it is seen that HKDF outperforms ECDH in terms of time and energy. This is because the HKDF makes use of only HMAC as compared to ECDH that makes use of expensive operations like scalar multiplication and radio communications. However, when considering security, ECDH outperforms HKDF. This is because the HKDF makes use of pre-loaded session key, as compared to ECDH, which generates the first session key through a Diffie-Hellman Exchange [134].

Table 5.4 presents the measurements for the AES-GCM 256 library in Contiki OS. The results show that this is approximately 80 times faster when compared to the AES-CBC 128 and AES-CBC 256 libraries in Wasp mote [18], [135]. This shows the superior technologies and techniques used in the presented system provide security with fast time and low current consumption.

Table 5.4: AES Analysis

<table>
<thead>
<tr>
<th>Platform ➔</th>
<th>libraries</th>
<th>AES (built-in Contiki)</th>
<th>AES (built-in Wasp mote)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics ↘</td>
<td>Total Time (ms)</td>
<td>0.132</td>
<td>10.44</td>
</tr>
<tr>
<td></td>
<td>Time for Decrypt stage (ms)</td>
<td>0.0752</td>
<td>3.21</td>
</tr>
<tr>
<td></td>
<td>Time for Encrypt stage (ms)</td>
<td>0.0569</td>
<td>7.23</td>
</tr>
<tr>
<td></td>
<td>Current (mA)</td>
<td>5.2</td>
<td>7.4 to 64.9</td>
</tr>
<tr>
<td></td>
<td>Notes</td>
<td>AES hardware accelerator</td>
<td>Hardware + Software</td>
</tr>
</tbody>
</table>

5.4 Pseudo Random Number Generator Seeding (PRNGS)

The previously mentioned key management schemes require a strong foundation. This may be achieved by strengthening the core random number generation process as it is used for initial random number generation at the BS [44] and for private key generation in ECDH. Therefore, in this system several techniques are discussed that can be used for the improvement of these processes. Following are the notations used in this section:

1) **RTIMER_NOW**: Current time in system ticks.
2) **srand**: 'C' library function from <stdlib.h>. This is used to initialise the PRNG with a seed.
3) **RAND_MAX**: Maximum value the random function (rand()) can return, and is equal to 2147483647.
Table 5.5 examines four scenarios with different initial seeding, and rand() function range. From the table, it is deduced that scenarios 3 and 4 are disregarded due to excessive time consumption (261ms), and low power-values variation respectively. However, Scenario 3 is still included in the following test for further comparison because it results in changes to output random numbers, visible to the naked eye, unlike Scenario 4.

Table 5.5: PRNGS Scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Initial seeding</th>
<th>rand() range</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>No seeding</td>
<td>0 to RAND_MAX</td>
<td>Results stored as characters, where every character corresponds to 1 byte</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>Using srand: srand( rand() % 256 )</td>
<td>0 to RTIMER_NOW</td>
<td>Results stored as characters, where every character corresponds to 1 byte.</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>Values (µ) from Accelerometer Sensor</td>
<td>0 to µ</td>
<td>Too much time consumption (261 ms)*</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>Power Values (β) obtained from Contiki’s powertrace [106]</td>
<td>0 to β</td>
<td>Power values do not vary considerably for the scenario to work</td>
</tr>
</tbody>
</table>

*Time taken is large as the sensor is calibrated using ‘adxl346.configure(ADXL346_CALIB_OFFSET, 0)’ function

Following test measures Scenarios 1, 2, and 3 using the NIST’s statistical test suite for random and pseudo random number generators [136]. This test suite consists of the following tests outlined briefly:

- **Frequency**: test shows whether the quantity of zeros and ones are roughly the same when compared to a true random sequence.
- **Block Frequency**: test takes as input values N, and n. Where N is the length of each block, and n is the length of bit string. The test shows if the number of ones in an N-bit block are approximately equal to N/2.
- **Cumulative Sum**: test checks whether there are excessive amounts of zeros and/or ones during the beginning stages of a sequence. This test focuses on the maximum absolute value of partial sums of the sequence.
- **Runs**: test takes as input an n bit string. The test shows whether the number of ‘uninterrupted identical bits sequence’ of zeros and ones of different lengths is similar to that in a random sequence.
- **Longest Run**: test shows whether the longest run of ones in a sequence is consistent with the longest run of ones for a random sequence.
- Rank: test is to verify whether linear dependence exists among a fixed length substrings of the sequence. It does so by calculating the rank of disjoint submatrices of the input sequence.
- FFT: test is the discrete Fourier Transform test. It is a spectral method that identifies periodic features in a bit series that indicate deviation from the normal i.e. randomness.
- Linear Complexity: test checks whether a sequence is complex. This is done as complexity adds to randomness. The test is focuses on linear feedback shift register.
- Time Consumed: test measures the amount of time it takes to generate a sequence.

In these tests, the number of input bit stream length \( n \) is 10000, and the significance level \( \alpha \) is set to 0.01, which means that 1 in every 1000 scenario may fail. Therefore, any proportion-value (\( P \)-value) \( \geq 0.01 \) means the PRNG has passed that statistical test. Table 5.6 shows the results with the \( P \)-value.

From the table, it is inferred that Scenario 2 performs better than the other scenarios even though it uses a PRNG. This is because the accelerometer in Scenario 3 has limited range and resolution, and the power consumption in Scenario 4 does not vary considerably, as mentioned previously. Moreover, the time consumed by Scenarios 1 and 2 is considerably less when compared to Scenario 3 (\( \approx 1/500 \)).

Additionally, Scenario 2 performs better compared to the Raw Data and Secure Hash algorithm approach used in [122]. However, it falls behind when compared to the
Washed+Rinsed technique and Blum-Blum-Shub scheme [122] as these schemes are oriented towards true random number generation.

5.5 Summary

The chapter combined different phases of the system to build towards a complete WSN solution for a patient monitoring system. It also incorporated security and key management techniques related to the WSN. These techniques are important because they prevent information leakage and various security attacks. This also affects the confidence of people involved with these systems. Therefore, the system phase presented in this chapter tackles the problem of security from different angles.

First, confidentiality and data-authentication were added using the AES-GCM 256-bit cipher, which outperforms other schemes as shown in Table 5.4. Second, key management schemes were included with session-key establishment, and key updates, which are supported by ECDH, and HKDF standards that outperform other presented schemes in Table 5.3. Additionally, key generation feature was incorporated. This is supported using the PRNG from Contiki’s ‘C’ library, which was analysed for different seeding scenarios using the NIST’s statistical test suite [136]. This provided approaches for seeding the PRNG to improve its randomness, thereby, giving more insight into the importance of random number generators for key-management.

To summarise, the system presented in this chapter provides a high level of security with internet connectivity for WSN devices. In addition, a working implementation on different technologies i.e. openmote, Intel Edison, Broker (Internet) and PC was presented, which is published in [137]. In the following chapter, the last phase of the system i.e. Cluster Elections and its security is presented to complete the system specification.
Chapter 6

Cluster Elections

Different mechanisms associated with the presented system have been described in Chapter 4 and Chapter 5. These mechanisms entail security for the routing network carrying patient data, and key management for the full system. These steps are important because reliability, security, and efficiency of the system increases confidence in its usage by the patients and the hospital staff, thereby allowing freedom of movement, ease of use, and emergency care.

This brings the focus onto the next and the last mechanism of the system i.e. the cluster of sensors represented on a patient’s body. Here the chapter discusses the cluster elections and the cluster’s security. Security is added to the cluster in form of data confidentiality, data authentication and entity authentication. And elections refer to the selection of a cluster head (CH), and rotation of the responsibility of CH among the sensors in the cluster. This rotation is quite important to achieve load balancing of the network, as it can help reduce the energy consumption of the network when coupled with other techniques [138]. Figure 6.1 shows the CH elections part of the presented system.

Figure 6.1: Cluster Head Elections. Entities involved are Router R, and cluster of sensors; security measures shown are ECDH and AES GCM 256
Here, the sensor nodes are in range with each other, and the router R. The communication between them is achieved using the RIME communication stack of the Contiki OS [49]. Finally, the security techniques used are AES GCM-256 for data confidentiality and data integrity at the sensors and the router R; and the ECDH algorithm for key management between the router R and the cluster of sensors.

The chapter is structured as follows: in Section 6.1 work related to cluster head elections in WSNs is described. In Section 6.2 the general layout/architectural structure of Medical Wireless Sensor Networks is discussed and the necessary equipment is outlined. In Section 6.3 experimental set-up is shown with focus on cluster security in Section 6.3.1. In section 6.3.2 the results related to Cluster elections and rotation are presented. Finally, in Section 6.4 this phase is discussed and concluded.

6.1 Current Solutions

Some of the work that is related to Cluster Head elections in WSN is as follows: In [139] the authors propose an improved version of Cluster head elections of the LEACH protocol [51]. Where they propose that the nodes do not send Advertising (ADV) messages often in the case when there are CHs in the vicinity. This allows a reduction in the number of CHs, and gives isolated nodes a chance to become a CH too. The protocol performs well compared to the traditional LEACH, but not as well when compared to their Integer Linear Program (ILP). In addition, the protocol is only implemented on a simulator, and not on hardware.

In [140] the authors propose a dynamic CH election protocol (DCHEP), which switches the CHs dynamically thereby providing network lifetime and availability. It outperforms its original standard by 26% in terms of energy consumption. The simulation of this protocol results in an energy consumption of 0.4J per application packet approximately, for each node in 1 to 100 nodes in the network. In comparison, the CH election presented in this chapter consumes 58.12mJ, on average, for each node in a network of 25 nodes (see Section 6.3.2).

In [141] the authors simulate a Stable Election Protocol (SEP), using TinyOS, for different topologies of various sizes. Where the SEP protocol uses weighed probabilities in relation to residual energy for becoming a cluster head. The average energy consumption for the scenario with 1 CH, and 25 member nodes is 1600mJ and 1000mJ respectively. This consumption decreases when there are more CHs, after that the energy consumption is seen to increase considerably, for e.g., when there are 50 member nodes, if the number of CHs are more than 2 then the energy consumption shoots up by 300-500
mJ for every CH added to the cluster. In contrast to this, our CH election process keeps only 1 CH at a time with a backup CH (i.e. Shadow), and the CH is rotated timely to balance the energy consumption, this energy consumption amounts to 58.12 mJ per node in the cluster.

Authors in [142] propose an energy efficient Virtual Cluster Head Election Scheme. The protocol is simulated on NS-2 simulator and compared against the LEACH protocol. Here the energy consumed by the protocol is lower than that consumed by LEACH, thereby increasing the lifetime of the WSN by 20%. The total energy consumption here is deduced from their graphs; this is approximately 75J after 200 seconds of operation. This amounts to a consumption of 6J after 16 seconds. This value is quite high as compared to the presented system: 1.45J for a network of 25 nodes for the time CH elections last i.e. 16 seconds (see Section 6.3.2). Moreover, the protocol cannot adapt to the scenario where nodes cannot reach the BS in one hop. In comparison, our protocol bundled with our routing protocol in Chapter 5 (published in [44]), can handle multi-hop connections from sensors to the BS via the APs or routers.

The authors in [143] implement cluster head elections specifically designed for energy and delay constraint WSNs. They do so by proposing a routing algorithm Energy Delay Index for Trade-Off (EDIT). Here the algorithm chooses the CH based on energy consumption and/or delay. The authors have in-depth comparison and results for energy and delay in the network, giving a good insight into the protocol and its mechanism. The average energy consumption per node for 50 nodes in the network is approximately 20J, per their graphs.

The work is also simulated and hence they can add large number of nodes. Unlike our work where we have implemented the protocols on hardware.

6.2 Architecture

In a WSN, the sensors comprise of wearable body sensors required to monitor the vitals of an individual. These sensors sense raw data like ECG, EEG, gait, locomotion and blood saturation levels. However, the raw data requires processing, which the sensors alone cannot provide. Therefore, each sensor has a corresponding node attached to it. This node is also responsible for communicating with other nodes and/or the router R.

Now, the data sensed/processed by these nodes is accumulated into the Cluster head (CH). The CH transmits aggregated data to the nearest router, router R. The router then relays data to the base-station which in-turn is responsible for forwarding it to the subscribed PC, figure 6.2.
CHAPTER 6. CLUSTER ELECTIONS

Here the BS first receives a command from the subscribed PC as described in Chapter 5. It forwards this command to router R to set-up a routing using the on-demand secure routing protocol in Chapter 4. This is followed by Router R broadcasting a ‘Hello’ packet to the sensors thus initiating the CH elections (see Appendix B.3 for important header file). Now, when the CH elections are initiated each sensor node calculates its weight based on the metrics described in Section 6.3, and broadcasts this weight into the cluster. Upon receiving each other’s weights, the sensor nodes select the CH as the node with the lowest weight, and a Shadow (S) i.e. backup-CH as a node with the second lowest weight.

Simultaneously key management (generation) is being enforced via ECDH to negotiate keys between R and cluster of sensors.

6.2.1 Platform and OS

Two platforms are used for the evaluation and validation of the CH Elections phase. Tmote-sky [84] and openmote-cc2538 [45], [107]. The former is only emulated on Contiki Cooja. It has been available for several years and therefore forms a good baseline for comparison. The latter is more recent, is powerful and has several useful additional features when compared to Tmote-Sky. More details about these platforms are described in Chapter 2.

Contiki OS is used as the operating system due to the reasons mentioned in Section 2.1.5. In addition, Contiki provides the following advantage for this phase of the system: Different radio sleep times can be enforced by different MAC layers and radio duty cycling (RDC) layers. For e.g., nullrdc provides no duty cycling at all, while ContikiMAC allows nodes to sleep 99% of the time [104]. This implies there is no need to implement a separate sleep-controlling interface, as implemented in [111]. This is advantageous because it
allows the sensors and the Router R to sleep most of the times when not exchanging radio messages.

### 6.3 Experimental set-up

In figure 6.2, wireless sensors are represented on a patient’s body to show the medical perspective of the system. However, in the implemented set-up these sensors are kept in the University Hallway (corridor) to sense raw data of the corridor room (*temperature, humidity and/or luminosity values*) whenever they are requested to do so by the subscribed PC. The reason for using room-sensed data instead of patient-sensed data is that no matter what type of data sensed by a sensor, the architecture of the system is such that the data stored and encrypted is send towards the base station in similar fashion. And since the conversion of raw data to stored sensed data, and study of different types of data is out of scope of this experiment, as mentioned in Section 2.1, therefore, the room-sensed values will suffice for the experiments conducted.

Now the primary goal of the sensors is to sense raw data and convert it to human readable form. However, this data must reach its destination safely. To facilitate these requirements, some measures are taken. First, it is not viable for all sensors to transmit data individually to the nearest Router i.e. Router R. This is because it may lead to loss of data through packet collisions and/or battery exhaustion. Therefore, these sensors elect a CH amongst themselves using a weight calculation procedure that uses the following metrics:

1) LQI and RSSI of the `Hello` packet broadcasted from R
2) Time spent by a node in each of its power states ($t_{ps}$)
3) Amount of the battery remaining (specific to Tmote-Sky only)
4) The number of times and the duration for which the node has previously served as a CH.

The above mentioned first and fourth points are well studied and therefore will not be elaborated here. Considering the second point i.e. the inclusion of $t_{ps}$: the time spent in the power-states is directly proportional to the energy consumption ($E_{ps}$) of the node as shown in Eq. 6.1.

$$E_{ps} = (Vcc \times Ic \times tps) \quad (6.1)$$

Regarding the third point, this method considers the `percentage of battery used` instead of `the amount of battery used irrespective of total battery charge`. This is advantageous when compared to the non-linear weight calculation in [111] because the latter may result
in a stagnant CH election process when the last node uses the first 20 percent of its battery. Thereby inhibiting the re-election of CH, and leading to excessive energy consumption for the previously elected CH.

It is obvious that the energy consumption is directly proportional to the battery consumption (\(C_c\)). But the remaining battery parameter (\(C_r = C_{total} - C_c\)) should not be included because it favours node(s) with larger battery capacities as in the linear weight calculation procedure presented in [111]. Hence, in the implemented system, a node is elected depending upon \(C_c\) only. This is advantageous because the method gives a fair chance to all nodes in becoming the CH, and is not biased towards a certain number of nodes that have larger power source. The disadvantage being that nodes with smaller battery source may die out faster.

6.3.1 Cluster Security

In this section the security associated with the cluster and its operations is discussed. The two categories that are included here are node addition and node deletion.

Node addition: If a new node is added to a cluster, this node sends a beacon into the cluster. This allows the nearest router (Router R) to re-initiate CH elections. The node addition feature is authenticated at R using an HMAC code. In this scheme the node to be added, say node \(x\) sends a `Hello` packet into the cluster, with the SHA-256 hash of the key (\(K_x\)) shared between Router R and the sensors. This HMAC is then verified by Router R. Once the HMAC is verified, R informs all the nodes in the cluster to re-initiate CH elections.

- Node \(x\) sends to the router R:
  - `Hello` || HMAC-HASH(\(K_x\))

- The AP upon receiving this message verifies
  - whether HMAC-HASH(\(K_x\)) is equal to HMAC-HASH\((K'_s)\), where \(K'_s\) is just the representation of key \(K_s\) stored on the AP.

- If the HMAC is verified
  - then the node is added into the Cluster, and CH elections re-initiated.

- Else
  - the connection is rejected

Node Deletion: If a CH or Shadow (S) is removed from the cluster, then router R identifies this through the absence of periodic beacons. If R confirms that CH or S is dead, it invokes the ECDH algorithm to generate a new session key (\(K_s\)) and re-initiates CH.
elections. This prevents the scenario where a node is captured, and can launch a variety of attacks due to a compromised session key. Now, if a normal node is removed from the cluster, the CH and S will note its absent periodic beacons. They will then remove the entry of this node from their Cluster Node ID List (CNIDL) list. Thereby allowing easy addition or/and removal of sensor nodes attached to a patient’s body.

6.3.2 Results

In this section the CH elections power measurements are presented. These measurements were taken using two different tools namely Contiki’s powertrace [106], and the Agilent 66321D Mobile communications DC source [108].

**Powertrace Power Calculations – Openmote and Tmote-Sky (Cooja)**

Powertrace tool provides a breakdown of different activities and components of the system in the form of raw values. It measures the amount of time spend by a node in each of its power states, and is accurate up to 94% for the Tmote-Sky platform [106]. However, the powertrace’s power values may differ from the measured current when divided by the applied voltage. This is because powertrace calculates the time spend in each power state every one second, which implies that the power is calculated for less than or equal to one second (max. 1Hz). However, an external source like Agilent 66321D can measure the corresponding current at much higher resolution (up to few microseconds).

The setup consists of three normal sensor nodes and an Access Point (AP) or router on two different platforms i.e. Tmote-Sky and openmote. Power was measured for source node with RIME node ID: 4, using two scenarios, where:

1) The radio duty cycling (RDC) layer is disabled (namely *nullrdc*) and the low power mode (LPM) is enabled, and

2) The RDC layer is enabled (*ContikiMAC*) and the LPM is disabled.
From the power measurements, it can be inferred that openmote and Tmote-Sky consume similar power for CH elections, with openmote being slightly more efficient. Figure 6.3 shows nullRDC + LPM scenario (Scenario 1) for both platforms. Here, the average power consumption of openmote is 60.743mW, and average power consumption of Tmote-Sky is 61.329mW. From these numbers, it is seen that the average power consumption of the openmote platform is 586μW (61.329 – 60.743) lower than the Tmote-sky platform.

Figure 6.3: Scenario 1 - Openmote and Tmote-Sky Average Power Consumption

Figure 6.4: Scenario 2: Openmote Power
Figure 6.4 shows the ContikiMAC + no LPM scenario (Scenario 2), which is implemented on openmote only. It contrasts the Central Processing Unit (CPU) power of openmote: 21.0044mW against its average power consumption: 21.5364mW. The corresponding energy values for the 16 seconds (from figure 6.6) that the CH elections last are 336.07mJ and 344.58mJ respectively.

Now if we consider Openmote specifically, Scenario 1, from figure 6.3 consumes an average current of 20.24mA (60.743 / 3V). While Scenario 2, from figure 6.4 consumes 7.17mA current (21.5364 / 3V). This means that a battery of 250mAh will last approximately 11.11 hrs for the former case and 31.3 hrs for the latter. Considering an excess of 10% to avoid deep discharging of the battery. Thus implying that Scenario 1 has 2.11 hrs better battery life than the scenario provided by the system in [111].

**Agilent 66321D DC Source Measurements - Openmote**

The Agilent 66321 DC source is a device used to measure the current of a node or other mobile devices in real time. It supplies the voltage required to the device, and calculates the current consumed. The device can sample as low as a 15 μS interval i.e. 66666.67 Hz [144] thereby showing its accuracy compared to powertrace; and is used in conjunction with 14565B device characterization software [109].

![Figure 6.5: Measured Current Consumption of Openmote for the different modes](image)

Figure 6.5 shows the current consumption of openmote for the previously mentioned two scenarios. NullRdc + LPM (Scenario 1) consumes an average current of 19.5465mA. While, the ContikiMAC + no LPM (Scenario 2) consumes an average current of
9.321mA. The corresponding energy values\(^9\) for the 16 seconds (from figure 6.6) that the CH elections last are 938.232mJ and 447.408mJ respectively.

This means that a battery of 250mAh will last approximately 11.51 hrs for the former case, and 24.14 hrs for the latter. Considering an excess of 10% to avoid deep discharging of the battery. Thus implying that Scenario 1 has 2.51 hrs better battery life than the scenario provided by the system in [111]. The reason for this battery performance gain is the fact that most of the processing is shifted towards the BS. Thereby, relieving the nodes from excessive processing.

**Simulation for Scalability – Tmote-Sky (Cooja)**

To better evaluate the scalability of implemented CH elections a network of nodes on the Cooja simulator (Contiki OS) is simulated. In this simulation 25 cluster nodes, 1 router, and 1 BS are deployed. The BS forms a path with the router, and the router initiates CH Elections in the cluster. The measurements were taken for the following two metrics:

1) Power consumption
2) CH rotation frequency

The former is measured using the previously mentioned powertrace tool, and the latter is measured by observing which node ID is the CH and which node is Shadow (S) for several iterations that are equal to the number of nodes: 25.

**Power Consumption**

![Average Power Consumption](image)

Figure 6.6: Average power consumption of randomly selected nodes from the cluster. After 16 seconds it is seen the elections stop due to same power values

Figure 6.6 shows the average power consumption per second for randomly selected nodes from a cluster of 25 nodes. The average power consumption over time for each

\(^9\) For when the voltage is set to 3V in the Agilent 66321D DC source
node is 3601.46 μW, 3711.88 μW, and 3598.94 μW for randomly selected node IDs 8, 13, and 20 respectively. This amounts to an average of 3637.42 μW resulting in an average energy consumption of 58.198mJ per node for the 16 seconds that the CH elections lasts. Hence, the average energy consumption of 25 nodes during the CH election is 1.45J. Thus showing an improvement by 0.342 J per node compared to [140], and 4.55 J for the whole cluster compared to [142].

After the elections are finished, the CH is responsible for aggregating encrypted data from the other sensors, and transmitting it to Router (R) from figure 6.1 or figure 6.2. However, this comes at a price of further power consumption. This is seen in figure 6.7, where the average power consumption is 4437.19 μW.

![Average Power Consumption](image)

Figure 6.7: Post-Elections CH Power Consumption. The spikes are due to radio receive/transmit.

**CH Rotation**

Moving on to the second category i.e. the CH rotation. It is invoked periodically to keep the load of the cluster balanced [145]. This is done because the CH consumes 799.77 μW more power (4437.19 – 3637.42) when compared to other nodes, and will therefore be under pressure most of the time.

For evaluation purposes the CH rotation is set to be invoked every 130 seconds, after which the `rotate_CH()` method is called (declaration in Appendix B.3). This method allows a node to re-elect CH and S based on the following metrics:

1) Charge: Overall energy consumption of a node (*lower is better*)
2) Duration_as_CH: Time for which a node has previously served as a CH (*lower is better*)
3) CH_count: Number of times a node has previously served as a CH (*lower is better*)

4) Battery_Level: The battery level of a node (*higher is better*)

5) RSSI\(^{10}\): Received Signal Strength Indicator from HELLO packet received from the nearest Router (*higher is better*)

6) LQI: Link Quality Indicator from HELLO packet received from the nearest Router (*lower is better*)

RSSI and LQI values indicate better communications strength/quality for high and low values respectively as these values corresponds to a strong signal with low noise [146].

Therefore, the CH is elected as the node ID with lowest node_weight in Eq. 6.2.

\[
\text{node_weight} = \text{charge} + (\text{dur_as_CH} + \text{CH_count}) - \text{Battery_Level} - \text{RSSI} + \text{LQI} \tag{6.2}
\]

where, \(\text{charge} = \text{avg_power} \times \text{time} \tag{6.3}\)

Here, the charge refers to the overall energy consumption up to the time at which the same charge is being calculated.

In Eq. 6.4, the average power (*avg_power*) is calculated for the Tmote-Sky platform emulated on the Contiki Cooja Simulator [99] using the powertrace tool mentioned previously.

\[
\text{avg_power} = (3L \times (\text{cpu} \times \text{DEC2FIX(cpu1,cpu2)} + \text{lpm} \times \text{DEC2FIX(lpm1,lpm2)} + \text{transmit} \times \text{DEC2FIX(TX1,TX2)} + \text{listen} \times \text{DEC2FIX(listen1,listen2)})/((64L \times \text{time})/1000)); \tag{6.4}
\]

Here, \(\text{cpu}\) is the time spent by the sensor node in its CPU state; \(\text{cpu1}\) and \(\text{cpu2}\) are the current measurements from Tmtoe-Sky’s datasheet\(^{11}\). Low power mode (\(\text{lpm}\)), TX (\(\text{transmit}\)), and RX (\(\text{listen}\)) modes have similar definitions. And, DEC2FIX is a macro pre-defined in the powertrace tool as follows:

\[
\text{#define DEC2FIX(h,d)} ((h \times 64L) + (\text{unsigned long})(d \times 64L) / 1000L)) \tag{6.5}
\]

\(^{10}\) Can be positive or negative value

\(^{11}\) http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf
Finally, in the following page, Table 6.1 shows the CH rotation results for the simulation of 20 nodes ranging from Node IDs 5 to 24. The columns consist of CH Node ID, Shadow (S) Node ID, CH weight, and S weight. The table shows, for every run, the Node ID elected as CH, the node ID elected as S and their corresponding node_weight values. The calculation of these weight values is already described at the beginning of Section 6.3.
Table 6.1: CH Rotation Frequency

<table>
<thead>
<tr>
<th>Run</th>
<th>CH Node ID</th>
<th>S Node ID</th>
<th>CH Weight</th>
<th>S Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>13</td>
<td>2386</td>
<td>2411</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>22</td>
<td>2625</td>
<td>2629</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>16</td>
<td>2661</td>
<td>2666</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>23</td>
<td>2712</td>
<td>2719</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>10</td>
<td>2739</td>
<td>2745</td>
</tr>
<tr>
<td>6</td>
<td>17</td>
<td>24</td>
<td>2767</td>
<td>2781</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>5</td>
<td>2797</td>
<td>2802</td>
</tr>
<tr>
<td>8</td>
<td>21</td>
<td>15</td>
<td>2816</td>
<td>2820</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>19</td>
<td>2856</td>
<td>2876</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>20</td>
<td>2895</td>
<td>3043</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>18</td>
<td>3082</td>
<td>3491</td>
</tr>
<tr>
<td>12</td>
<td>22</td>
<td>6</td>
<td>3509</td>
<td>3532</td>
</tr>
<tr>
<td>13</td>
<td>24</td>
<td>16</td>
<td>3163</td>
<td>3551</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>14</td>
<td>3582</td>
<td>3592</td>
</tr>
<tr>
<td>15</td>
<td>23</td>
<td>10</td>
<td>3618</td>
<td>3630</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>9</td>
<td>3652</td>
<td>3668</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>21</td>
<td>3170</td>
<td>3175</td>
</tr>
<tr>
<td>18</td>
<td>15</td>
<td>12</td>
<td>3193</td>
<td>3215</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>8</td>
<td>3244</td>
<td>3254</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>13</td>
<td>3416</td>
<td>3441</td>
</tr>
<tr>
<td>21</td>
<td>18</td>
<td>22</td>
<td>3854</td>
<td>3868</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>6</td>
<td>3905</td>
<td>3905</td>
</tr>
<tr>
<td>23</td>
<td>14</td>
<td>23</td>
<td>3951</td>
<td>3963</td>
</tr>
<tr>
<td>24</td>
<td>7</td>
<td>17</td>
<td>3978</td>
<td>3987</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>5</td>
<td>4003</td>
<td>4022</td>
</tr>
</tbody>
</table>

*First repetition of CH and S node ID combination
From the results, it can be deduced that eventually every node in the cluster gets a chance to be a CH, and/or S. So, for \( n \) nodes in a cluster, there are \( n \) unique CHs, and at least \( n/2 \) unique CH and S combinations. These CH and S combinations can be increased depending upon the weight resulting from node calculation procedure \((\text{node\_weight})\):

\[
P_2 \text{ permutations, where } n \text{ is the number of nodes in a cluster} \quad (6.6)
\]

From Eq. 6.6, for a cluster containing 20 nodes in Table 6.1, there will be \( 20P_2 \) unique possibilities of CH and S combinations over time. This amounts to 380. However, depending on the \( \text{node\_weight} \) there will be repetitions, but looking at Table 6.1 it is seen that there will be no repetitions if `rotate_CH()` is invoked \(< n \) times. This is acceptable because in real-world implementation there should not be any need to rotate the CH this frequently.

6.4 Summary

In this chapter the Cluster head elections and security were outlined and discussed in detail. This phase was implemented on real-time hardware openmote, and also simulated on the Tmote-Sky platform in Contiki Cooja simulator. This implementation included election of a cluster head amongst the cluster of sensors on a patient’s body, followed by the rotation of the cluster head to other sensors to keep the network load-balanced. The results arising were measured against different solutions from Section 6.1 [139]–[143]. Compared to these solutions the CH elections implemented in this chapter perform better in terms of energy and rotation frequency. These results are published in [147], [128]. This phase is an important part of the complete system because it allows the sensors to sense data from the patient, while maintaining the cluster, and maintaining network connectivity with a router. It also simplifies network management by keeping the cluster self-organising and self-healing.
Chapter 7

Conclusion

In this thesis, a secure end-to-end wireless sensor network solution was implemented for medical patient monitoring purposes. The major problem motivating this thesis arises from the lack of complete end-to-end systems and security solutions deployed for WSN, which is due to the nature of wireless networks and lack of research in sensor security by industry. This leads to sensors and/or networks with minimal or no security at all. There have been solutions for full systems and security, reviewed throughout the thesis, but either they are implemented completely on simulator or they focus only on parts of the system. In contrast to this, the presented system achieves its objectives:

1) DoS resistance: against BH and SF attacks achieved via hashing scheme and neighbour monitoring mechanisms respectively. Thereby fortifying security against any future BH / SF attacks on WSN systems.

2) Entity authentication: achieved for Subscribed PC, and Gateway via TLS and username and password.

3) Data confidentiality and Authentication: achieved via AES-GCM 256.

4) Energy/Power optimisation:
   a. by deploying a load balanced cluster using cluster elections and rotation.
   b. By using ContikiRDC layer for routing and sensor nodes to keep Energy/Power consumption lower compared to when there is no radio duty cycling.

5) Constructing the whole system: Secure on-demand routing, implementation of ECDH and HKDF key management, cluster security and elections, applications for Gateway and subscribed PC, and modifications to the lower layers of Contiki OS to achieve a fully functional WSN system for patient monitoring.

Following paragraphs show the breakdown of the objectives achieved by the presented system according to the chapter in which they are described:

Chapter 2 reviewed the field of WSN to build a foundation in understanding the system presented throughout the thesis. This chapter included existing systems on WSN for healthcare application, details on other applications, protocols, hardware platforms, and operating systems.
Chapter 3 reviewed the security in WSNs with focus on an outline of the presented system. Security for each phase of this system is explained in its respective chapter with a discussion on system and security throughout the thesis. This gives a good outline of the full system before the reader gets into details.

Chapter 4 discusses the secure routing phase between the BS and end router. This phase fortifies an on-demand routing protocol against DoS attacks like black-hole and selective forwarding. This is useful because it prevents the system from failing. The phase was deployed on both simulator and hardware, and tested extensively.

Chapter 5 implements key management techniques for the presented system. These techniques were studied extensively and compared with other systems. The resulting solution of HKDF and ECDH were deployed between PC and Router, and Router and cluster of sensors respectively. This work improves the implemented security features by managing and updating keys periodically. Additionally, the chapter provides seeding techniques for PRNG, which were compared and tested using the NIST statistical suite for random number generators.

Chapter 6 introduces the Cluster Elections process with cluster security. This phase allows safe node addition and deletion, and helps balance the network load in the cluster of sensors by electing and rotating the cluster head amongst the sensor nodes. The phase was deployed on both simulator and hardware, and tested extensively.

Finally, the full system comes together in this chapter. Here the system is evaluated by comparing the results for hardware and simulator to fortify the research undertaken. This is done to show the usability and feasibility of the system. Additionally, this chapter concludes the thesis by highlighting the contributions made, and future work possibilities.

The rest of the chapter is structured as follows: in Section 7.1, the full end-to-end WSN system for patient monitoring is implemented in the hallway/corridor of University of Limerick. In Section 7.2, Simulation Vs Hardware results are discussed with focus on Secure routing in Section 7.2.1, and Cluster Elections in Section 7.2.2. In Section 7.3, the contributions made by the research in this thesis are presented. Finally in Section 7.4 possible future work directions are given.

### 7.1 System Implementation

This section presents a discussion of the complete system that has been described component by component in the previous chapters. The system was implemented, and tested through the corridors of the University of Limerick to mimic a hospital ward environment where the routing nodes may be in different halls/rooms (*ward*), and the
Subscribed PC (*Hospital Staff’s PC*) can be anywhere in the hospital or even outside of it. The model is represented in figure 7.1.

![Diagram of System Deployment](image)

**Figure 7.1:** System deployment at the university (UL) hallway

The process flow is as follows:

1) The subscribed PC sends *start* command to the gateway via the *mosquitto* broker.

2) The Gateway injects this command into the wireless sensor network (WSN) by sending it to the BS via UART.

3) The BS initiates the secure routing protocol (*see* Chapter 4) to form an on-demand path with the source router R.

4) Once the connection is verified, Router R initiates ECDH (*see* Chapter 5) between itself and the Cluster of sensors while simultaneously starting the cluster head elections (*see* Chapter 6).

5) Once sensors have data, they encrypt it and transmit it to their Cluster Head (*see* Chapter 6), which aggregates encrypted data from all sensors, and sends it on to the router.

6) This encrypted data is decrypted at the router with session key $K_s$ (Section 5.3.1), and then re-encrypted with another key $K_m$.

7) This encrypted data is then forwarded to the Subscribed PC via BS and Gateway. In this process, the subscribed PC periodically updates the session keys ($K_m$) between itself and router R using HKDF (Section 5.3.1). This can also be achieved at the user’s will by clicking on the *update* command in the GUI presented in Figure 7.2. Additionally,
the user may cease all operations of the network by clicking the *stop* command from the same GUI window.

![GUI running on the Subscribed PC. The commands available are start, stop, and update. The window also displays the process and current network set-up with Sequence numbers for the secure routing protocol.](image)

Following steps show the handling of encryption and decryption of data:

- The cluster head of Sensors send data to Router (R)
  - \( \{ \text{Data} \} K_{r-ch} \)
- (R) decrypts the Encrypted data
  - \( \{ \{ \text{Data} \} K_{r-ch} \} K_{r-ch} = \text{Data} \)
- (R) encrypts data
  - \( \{ \text{Data} \} K_{pc-r} \)
- (R) forwards the data into the network, towards the PC
- PC receives encrypted data, and decrypts it
  - \( \{ \{ \text{Data} \} K_{pc-r} \} K_{pc-r} = \text{Data} \)

Where, \( K_{r-ch} \) is the AES-GCM 256 session key between Router (R) and the cluster of sensors; \( K_{pc-r} \) is the AES-GCM 256 session key between the subscribed PC and Router (R). The former key is generated using the ECDH algorithm, while the latter is generated using the HKDF algorithm.

### 7.2 Simulation Vs Hardware

This section highlights the results that achieved on both simulator and hardware for Secure routing (Chapter 4) and Cluster Elections (Chapter 6) phase only. This is because the key management phase was implemented on hardware only due to unavailability of Openmote on Cooja Simulator, and emulation of Openmote’s Elliptic curves and SHA Hardware accelerators on the Cooja Simulator. The section shows a comparison between
simulator platform and hardware platform in terms of the current consumed by a single
sensor node. The difference in this comparison justifies the usage of both simulator and
hardware for covering the dynamics of the implemented system. Additionally, the section
also shows the difference between measurements taken from the power measurement tool
and the current measurement device i.e. Powertrace tool [106], and Aglient 66321D
Mobile DC Source [144] respectively.

The simulator used, as mentioned before in Chapter 4 and Chapter 6, is Contiki Cooja
simulator that is capable of emulating a hardware node namely Tmote-Sky. The current
values obtained for this platform are calculated by:

1) Dividing the power values obtained using Powertrace tool with the input voltage
of Tmote-Sky i.e. 3V [84].

The hardware used, as mentioned throughout the thesis, is Openmote. The current values
obtained for this platform are calculated using two ways:

1) Dividing the power values obtained using Powertrace tool with the input voltage
of Openmote i.e. 3V [121], and

2) Using Agilent 66321D Mobile DC Source.

7.2.1 Secure Routing

This section compares the results obtained for the secure routing phase in Section 4.4.3
and Section 4.4.4.

Table 7.1 shows the comparison of these results. The first column shows the Tmote-Sky
(Simulated) power values obtained from Section 4.4.3, the second column shows the
Tmote-Sky (Simulated) current values obtained by dividing the power values with
Tmote-sky voltage i.e. 3V, and the third column shows the Openmote (Hardware) current

<table>
<thead>
<tr>
<th></th>
<th>Tmote-Sky (Simulated) Power</th>
<th>Tmote-Sky (Simulated) Current</th>
<th>Openmote (Hardware) Current</th>
<th>Difference in current abs(C1-C2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Phase</td>
<td>62.780mW</td>
<td>20.926mA</td>
<td>20.223mA</td>
<td>0.703mA</td>
</tr>
<tr>
<td>Detection Phase</td>
<td>62.932mW</td>
<td>20.977mA</td>
<td>21.0579mA</td>
<td>0.0809mA</td>
</tr>
</tbody>
</table>

1 Obtained using Powertrace tool
2 Current derived by dividing power values with Tmote-Sky voltage 3V
3 Current measured using Agilent 66321D Mobile DC source
4 abs (C1-C2) refers to absolute value i.e. it converts a negative result to positive result

Table 7.1 shows the comparison of these results. The first column shows the Tmote-Sky
(Simulated) power values obtained from Section 4.4.3, the second column shows the
Tmote-Sky (Simulated) current values obtained by dividing the power values with
Tmote-sky voltage i.e. 3V, and the third column shows the Openmote (Hardware) current
values obtained from Section 4.4.4. The final column shows the difference between the current consumption of both platforms.

The results are different because:

1) Emulation of Tmote-Sky on the Cooja simulator is different from the Openmote hardware being used (due to unavailability of Openmote emulation on Cooja), and

2) Measurement using Agilent 66321D includes an overhead of switching ON a LED on Openmote. This is included because current measurement is conducted manually by a user when they click the start button on the PC software for Agilent (14565B device characterisation software). Therefore, for a user to know when to start this measurement for a particular phase, the LED is switched ON by the program on Openmote.

7.2.2 Cluster Elections

Table 7.2 shows the comparison of Cluster elections phase implemented on cooja simulator for Tmote-Sky platform with the same phase implemented on Openmote hardware. The first column shows the Tmote-Sky (Simulated) power values obtained using Contiki’s Powertrace tool from Section 6.3.2, the second column shows the Tmote-Sky (Simulated) current values obtained by dividing the power values with Tmote-Sky voltage i.e. 3V, and the third column shows the Openmote (Hardware) current values obtained using Agilent 66321D mobile DC source from Section 6.3.2. The final column shows the difference between the current consumption of simulator and hardware platforms.

<table>
<thead>
<tr>
<th>Modes</th>
<th>Tmote-Sky (Simulated) Power¹</th>
<th>Tmote-Sky (Simulated) Current² (C1)</th>
<th>Openmote (Hardware) Current³ (C2)</th>
<th>Difference in current abs(C1 – C2)⁴</th>
</tr>
</thead>
<tbody>
<tr>
<td>nullRDC + LPM</td>
<td>61.329mW</td>
<td>20.443mA</td>
<td>19.5465mA</td>
<td>0.8965mA</td>
</tr>
</tbody>
</table>

¹Obtained using Powertrace tool
²Current derived by dividing power values with Tmote-Sky voltage 3V
³Current measured using Agilent 66321D Mobile DC source
⁴abs (C1-C2) refers to absolute value i.e. it converts a negative result to positive result

The results are slightly different because:

1) Emulation of Tmote-Sky on the Cooja simulator is different from the Openmote hardware being used (due to unavailability of Openmote emulation on Cooja), and

2) Measurement using Agilent 66321D includes an overhead of switching ON a LED on Openmote. This is included because current measurement is conducted...
manually by a user when they click the start button on the PC software for Agilent (14565B device characterisation software). Therefore, for a user to know when to start this measurement for a particular phase, the LED is switched ON by the program on Openmote.

Table 7.3 shows the comparison of tool and device used on Openmote (Hardware) only. The first column shows the Openmote power values obtained using Contiki’s Powertrace tool from Section 6.3.2, the second column shows the Openmote current values obtained by dividing the power values with Openmote voltage i.e. 3V, and the third column shows the Openmote current values obtained using Agilent 66321D mobile DC source from Section 6.3.2. The final column shows the difference between the current consumption measured using Powertrace and Agilent.

The results obtained compare the difference between Powertrace tool, and the Agilent 66321D mobile DC source measuring CH elections result on Openmote. The results are different because:

1) Measurement for power values using Powertrace tool is 94% accurate [106], and
2) Measurement using Agilent 66321D includes an overhead of switching ON a LED on Openmote. This is included because current measurement is conducted manually by a user when they click the start button on the PC software for Agilent (14565B device characterisation software). Therefore, for a user to know when to start this measurement for a particular phase, the LED is switched ON by the program on Openmote.

From this discussion it is seen that testing the research undertaken on both simulator and hardware is a good way to fortify it because one alone (Simulator or Hardware) may not be able to measure the full functionality of the presented work.
7.3 Contributions

Following the full functionality of the presented system, the contributions made by the same are outlined as follows:

1) The major contribution is the modification of an on-demand routing protocol to integrate security into it. This is because:
   a. It enables higher security in WSN systems.
   b. Increases the confidence of the patients that may use this system in the future.
   c. Improves reliability of the system from the hospital staff’s perspective.
   d. It can be adapted to other application systems like marine, civil etc.

2) The research presents key management features for different parts of the system. These features are implemented using HKDF and ECDH. The latter is not known to be used much in sensor networks especially across different platforms (PC <-> Sensor node router). However, it improves security considerably while not consuming large space due to the usage of a 256-bit Elliptic Curve.

3) The research implements a Cluster Elections phase that helps balance the network load by electing and rotating the cluster head.

4) The thesis provides a detailed study on several components of wireless sensor networks:
   a. It reviews the different applications, protocols and platforms that relate to the WSN. This is useful for any researcher looking for a quick comparison that could help them in making informed decisions about their own work.
   b. It provides detailed descriptions of the available and latest operating systems for sensor nodes. This is useful because the new operating systems are compared for the benefit of the implemented system and any other interested researcher.
   c. Finally, the thesis also reviews the different security attacks and solutions that can help improve other systems in addition to the presented system.

5) Finally, most of the implementations and experiments were conducted on both simulation environment and hardware environment. This improves confidence in the results, and allows other researchers to get a comprehensive experiment set-up all in one place.

The contributions mentioned here are published in the following Journals, book chapter and/or conferences:
• Comparison and overview of WSN in [16]
• Cluster Elections phase in [110], [127], [128]
• Secure Routing phase in [44], [110]
• Secure Key Management phase in [137]12
• Full System in [148]13

The work from these papers is cited in 17 different papers according to Researchgate database14.

7.4 Future Work

The different chapters of this thesis formulate an end-to-end WSN solution for a patient monitoring system. This system encompasses a wide range of topics and solutions for its different components and/or phases.

Even though major work has been already done in each phase of the system, there are several possibilities of expanding the work:

1) Secure Routing: Chapter 4 secures an AODV routing protocol against black-hole and selective forwarding DoS attacks.
   a. In the case of the former, the defense mechanism presented in Section 4.2.3 implements BS authentication using hash functions incorporated with RREQ packet. However, it does not implement the same with RREP packet because forging a RREP packet can only launch a bogus data sending attack from the malicious node to the BS. Therefore, in the future the inclusion of hash with RREP packet could be added to improve the security. However, the researcher must optimise the associated energy consumption also.
   b. The latter i.e. selective forwarding mechanism is comprehensive and does not leave much for future work except reducing the power/energy consumption.
   c. Finally, this secure protocol can be expanded to include security against other routing attacks like wormhole/sinkhole, and Sybil attack to fortify the routing process. The latter was attempted on the Contiki Cooja...
simulator using Hidden Markov Models but it requires more investment in terms of time and research.

2) Key Management: Chapter 5 presents key management features HKDF and ECDH implemented in two different parts of the system.
   a. In this case, the future work could include the expansion of ECDH over the complete system due to its superior security features. This must take into account exchanging information, for Diffie-Hellman exchange, across multiple platforms (PC, Gateway \(\leftrightarrow\) Router nodes), which will require considerable amount of time and research.
   b. Additionally, the PRNGS can be expanded to include more scenarios relating to the seeding of the generator.

3) CH Elections: Chapter 6 presents the CH elections and rotation scheme that is implemented for the cluster of sensors that are attached to a patient’s body.
   a. The scheme uses some important metrics for the election of Cluster head. These are packet RSSI, LQI, \(t_{ps}\), remaining battery, CH count and CH duration. Currently the scenario for RSSI and LQI combination considers low noise situations only. In the future, this may be expanded to consider noise.
   b. This phase includes node authentication and key revocation during the addition of a new sensor node. It also revokes keys during the deletion of a CH or S, however not during the deletion of a sensor node as it causes problems to the ongoing data communications. Therefore, the inclusion of additional security here would be valuable for future work.

Apart from the phases of the system, there are some other possibilities for expanding the work:

1) Expanding the GUI application on Linux-PC to incorporate complete network management i.e. inclusion of the health of every router node, and sensor node. The knowledge gained from this work may be used further to:

   1) Research more into DoS attacks for WSNs using the study published on secure routing.
   2) Incorporate other security mechanisms into the Contiki OS using the modification of RDC layer in Section 4.3 as a reference.
Appendix A

Publications

Journal Articles and Book Chapters


Conferences


M. Rao, T. Newe, I. Grout, E. Lewis, and A. Mathur "FPGA based Real time secure body temperature monitoring suitable for WBSN", in *International Conference on Computer and Information Technology (CIT)*, 2015. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.22
Appendix B

Programming Code (highlights)

B.1 Linux-PC Application (main function)

The fragment of code shows the main function of the Linux PC GUI application. This function shows a TLS connection with the mosquitto broker running on the same computer for testing purposes.

```python
organization = "ok9ofr"
deviceType = "Linux-PC"
username = "mosquitto_mqtt"
password = "torque@mqtt$7"

macAddress = hex(uuid.getnode())[2:-1]
macAddress = format(long(macAddress, 16), '012x')

# Generate Client ID and Broker address
clientID = "d:" + organization + ":" + deviceType + ":" + macAddress
broker = "localhost"
client_global = paho.Client(clientID)
client_global.username_pw_set(username, password)
client_global.on_connect = on_connect
client_global.on_message = on_message

# Set the appropriate certificates
cia_certs = "/home/avi/phdsys/TLS/local_cert.pem"
certfile = "/home/avi/phdsys/TLS/subscribed_pc.crt"
keyfile = "/home/avi/phdsys/TLS/subscribed_pc.key"
cert_reqs = ssl.CERT_REQUIRED
tls_version=ssl.PROTOCOL_TLSv1

# MQTT Connect Securely
client_global.tls_set(ca_certs, certfile, keyfile, cert_reqs, tls_version, None)
client_global.connect(broker, 8883, 60)
client_global.loop_start()
gui_construct()
```
print('Sampling data from Openmote BS...')
while True:
    if cmd_recvd == "stop":
        cmd_recvd = ""
        return 0;
    dr = ser.read(1)

    # Check for Duplicate frames in real time
    if s_flag == True and dr != '#' and dr != '/':
        data_recvd += dr
    if '#' in prev_dr and '#' in dr:
        s_flag = False
        send_data = True
    if '/' in prev_dr and '/' in dr:
        s_flag = True
    prev_dr = dr

    if s_flag == False and send_data == True:
        dataB_str = ""
        dataB_str = convert_data(data_recvd)

        # Send to cloud using MQTT
        client.publish(publish_topic, payload=dataB_str, qos=0, retain=False)
        print('Data published to Linux-PC')
        data_recvd = ""
        send_data = False
B.3 ClusterScheme.h (Sensor nodes)

```c
#ifndef ClusterSchem_H
#define ClusterSchem_H

#include "net/rime/rime.h"
#include "net/linkaddr.h"
#include "sys/ctimer.h"
#include "bodysensor.h"
#include "share_variables.h"

#define PERIODIC_BEACON 0

/*Type of Nodes*/
struct entities {
    linkaddr_t ID;
    uint32_t weight;
};

/*List to hold all of the Cluster Heads*/
struct sys_lists {
    linkaddr_t ID;
    uint8_t beacon;
    uint32_t weight;
    uint8_t count;
};

/*Structure to hold the address and weight of a node*/
struct weight_msg {
    linkaddr_t addr;
    uint32_t weight;
};

/*Beacon structure*/
struct beacon_msg {
    linkaddr_t addr;
};
```

123
uint8_t type;
}

struct entities CH; /*ClusterHead*/
struct entities S; /*Shadow CH*/
struct entities RTR; /*Router*/
struct entities node; /*Sensor Node*/

struct sys_lists CNIDL[MAX_NO_NODES]; /*Cluster Node ID List*/
struct sys_lists CHIDL[MAX_LIMIT_CHIDL]; /*Cluster Head ID List*/
struct sys_lists RIDL[MAX_RTR]; /*Router ID List*/

void CH_election(void);
void Calculate_set_Weight(uint32_t, uint16_t);
void router_selection(uint16_t);
void set_received_weights(const linkaddr_t *, uint32_t);
void clear_cnidl(void);
void clear_chidl(void);
void clear_lists(void);

uint8_t delete_from_cnidl(uint8_t, uint8_t);

uint32_t CalculateCharge(void);

/*Timer function used by a node to send beacons into the cluster*/
void beacon_timer1(void *);

/*Timer function used by a CH to send beacon to its Router*/
void beacon_timerCH(void *);

/*Timer function used to declare the status of a node in nodestatus_pr Process*/
void zero_timer( void* );

/*Function that decides whether the node is dead or alive*/
void node_status(void *);
/*Function that decides whether the CH is dead or alive*/
void CH_status(void *);

/*Function used to post the election Process*/
void post_election(void *);

/*Function that sends the Cluster Node ID List*/
void send_CNIDL(void);

/*Function that rotates the Cluster Head*/
void rotate_CH(void *);

/*Supporting variables for rotate_CH(void *)*/
uint32_t time_since_CH;
uint32_t dur_as_CH;
uint32_t charge;

linkaddr_t old_CH; /*Previous CH*/
linkaddr_t old_S; /*Previous Shadow*/

#endif

B.4 Keys.h (Router nodes)
#ifdef KEYS_H_
define KEYS_H_

PROCESS_NAME(gcm_process);
PROCESS_NAME(ecdh_pr);

#include "dev/gcm.h"
#include "mesh.h"
#define SIZE_EC8
#define SEND_SIZE_ARR (SIZE_EC + SIZE_EC + 1 + 1)
#define RECV_SIZE_ARR (SIZE_EC + SIZE_EC + 1)

/\*Router <-> PC Side: HKDF\*/
/\*This is the encryption key and the input keying material (IKM) for first periodic update\*/
uint8_t Km[KEY_SIZE+1];

/\*Session key after first periodic update\*/
uint8_t Ks[KEY_SIZE];

uint8_t encrypted_data[MAX_PAYLOAD_SIZE+GCM_TAG_LEN];
uint8_t encr_data_to_send[MAX_PAYLOAD_SIZE+GCM_TAG_LEN];
uint8_t encr_datalen;
uint8_t input_datalen;

/\*Router <-> Sensor side: ECDH\*/
typedef struct{
    uint32_t x[12];
    uint32_t y[12];
} ecc_point;

ecc_point Qr; \/*Router Public Key\*/
ecc_point recvd_Qs; \/*Received Sensors’ Public Key\*/
ecc_point ss; \/*Shared Secret\*/

process_event_t Qs_received;

void init_keys();

#endif
Bibliography


2002.


