

ULRR

On the security and energy consumption estimation of wireless sensor network protocols

Item Type	Thesis
Authors	Zhang, Fan
Download date	2026-03-16 21:10:04
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/2857

On the Security and Energy Consumption Estimation of Wireless Sensor Network Protocols

Author: Fan Zhang (MEng)

ID: 0526401

Supervisors: Dr. Reiner Dojen

Prof. Tom Coffey

Department of Electronics and Computer Engineering

University of Limerick

Ireland

5 November 2012

Submitted to the University of Limerick

in requirement of Doctor of Philosophy

Abstract

Along with the recent rapid development of Wireless Sensor Network (WSN) systems, the range of attacks against WSN routing protocols have grown. As a result, there is an increased need for secure WSN routing protocols. WSN routing protocols should be secured once they are involved in sensitive data transmission. However, secure routing protocols require extra time and energy for security computations. Further, due to the limited power supply of WSN nodes, it is useful to theoretically evaluate the energy consumption of the protocol prior to their deployment.

This thesis concerns the energy efficiency of WSN routing protocols and the cost of security in terms of additional energy dissipation. Two new secure routing protocols are proposed and their security is analyzed. Further, the energy consumption of the new protocols is evaluated using estimation methods and empirical measurements. Comparison of these results reveals that current estimation models have significant inaccuracies. A new energy consumption estimation model is proposed that computes the energy consumption of microcontrollers during: security operations, memory read-write operations and radio transceiver actions. Applying the proposed estimation model against the previously analyzed protocols demonstrates that it more accurately forecasts the actual energy consumption than traditional models.

Declaration

This thesis is entirely my own work and has not been submitted to any other university of higher education institute, or for any other academic award in this university. Where use has been made of the work of other people, it has been fully acknowledged and referenced.

Signature:

Fan Zhang

5 November 2012

Acknowledgements

I would like to thank everyone who has helped me through this project including my supervisor Prof. Tom Coffey and Dr. Reiner Dojen who always gave me good advice, guidance and directions.

Contents

ABSTRACT	I
DECLARATION	II
ACKNOWLEDGEMENTS	III
CONTENTS.....	IV
LIST OF FIGURES.....	X
LIST OF TABLES.....	XII
GLOSSARY	XIII
PART 1.	
1. INTRODUCTION	1
1.1 GENERAL INTRODUCTION.....	1
1.2 NECESSITY OF SECURITY PROTECTIONS TO WSN ROUTING PROTOCOLS AND ITS PRICE.....	2
1.3 REQUIREMENTS OF ENERGY CONSUMPTION ESTIMATION.....	3
1.4 OBJECTIVE OF THE THESIS	4
1.5 STRUCTURE OF THE THESIS.....	5
2. WIRELESS SENSOR NETWORKS	7
2.1 WIRELESS SENSOR NETWORK DEFINITION.....	7
2.2 WIRELESS SENSOR NODE	8
2.3 THE CHARACTERISTICS OF WIRELESS SENSOR NETWORK	9
2.3.1 <i>Large Scale Network</i>	9
2.3.2 <i>Self-organized Network</i>	10
2.3.3 <i>Dynamic Network</i>	10
2.3.4 <i>Application Oriented Network</i>	11
2.3.5 <i>Data Centric Network</i>	11
2.3.6 <i>WSN and Mobile Ad-hoc Networks Characteristics Comparison</i>	11
2.4 LIMITATIONS OF WSN NODES.....	13
2.4.1 <i>Limited Power Supply</i>	13
2.4.2 <i>Limited Communication Capability</i>	13
2.4.3 <i>Limited Computation and Storage Capacity</i>	14

2.5	ENERGY CONSUMPTION OF WIRELESS SENSOR NETWORKS	14
2.5.1	<i>The Importance of Energy Conservation</i>	15
2.6	APPLICATION DOMAINS OF WIRELESS SENSOR NETWORKS	16
2.6.1	<i>Military Application Area</i>	16
2.6.2	<i>Environmental Application Area</i>	18
2.6.3	<i>Health Application Area</i>	19
2.6.4	<i>Commercial Application Area</i>	20
3.	REVIEW OF WIRELESS SENSOR NETWORK ROUTING PROTOCOLS	21
3.1	MULTI-PATH ROUTING AND CLUSTER-BASED ROUTING	21
3.2	LOW ENERGY ADAPTIVE CLUSTERING HIERARCHY (LEACH)	23
3.3	POWER-EFFICIENT GATHERING IN SENSOR INFORMATION SYSTEMS (PEGASIS)	25
3.4	BASE-STATION CONTROLLED DYNAMIC CLUSTERING PROTOCOL (BCDCP)	26
3.5	CLUSTER BASED KEY MANAGEMENT PROTOCOL (CBKM)	28
3.6	SECLEACH	32
4.	SECURITY OF WIRELESS SENSOR NETWORKS	36
4.1	SECURITY REQUIREMENTS OF WSN ROUTING PROTOCOLS	36
4.1.1	<i>Availability</i>	36
4.1.2	<i>Integrity</i>	37
4.1.3	<i>Confidentiality</i>	37
4.1.4	<i>Authenticity</i>	38
4.1.5	<i>Freshness</i>	38
4.2	ATTACK AGAINST WIRELESS SENSOR NETWORKS	39
4.2.1	<i>Attacks Breaching Confidentiality</i>	39
4.2.2	<i>Attacks Breaching Integrity</i>	40
4.2.3	<i>Attacks Breaching Freshness</i>	40
4.2.4	<i>Attacks Breaching Authenticity</i>	40
4.2.5	<i>Typical Attacks</i>	41
4.3	SECURITY ANALYSIS OF THE BCDCP PROTOCOL	44
4.3.1	<i>Attacks Breaching Confidentiality of BCDCP</i>	45
4.3.2	<i>Attacks Breaching Integrity of BCDCP</i>	45
4.3.3	<i>Attacks Breaching Freshness of BCDCP</i>	46
4.3.4	<i>Attacks Breaching Authenticity of BCDCP</i>	47
5.	PROTECTING WIRELESS SENSOR NETWORK ROUTING PROTOCOLS	48

5.1	COUNTERMEASURES TO THE ATTACKS AGAINST WSN ROUTING PROTOCOLS	48
5.1.1	<i>Countermeasure to Attacks Breaching Confidentiality</i>	49
5.1.2	<i>Countermeasure to Attacks Breaching Integrity</i>	50
5.1.3	<i>Countermeasure to Attacks Breaching Freshness</i>	50
5.1.4	<i>Countermeasure to the Attack Breaching Authentication</i>	52
5.2	BLOCK CIPHERS	52
5.2.1	RC5	52
5.2.2	<i>Advanced Encryption Standard (AES)</i>	54
5.2.3	<i>Block Cipher Modes of Operations</i>	56
5.3	MESSAGE AUTHENTICATION CODE	58
5.3.1	<i>Hash-based Message Authentication Code</i>	59
5.3.2	<i>Cipher-based Message Authentication Code</i>	60
5.4	SECURITY PROTECTION'S IMPACT TO WSN NODE ENERGY CONSUMPTION	62
5.5	THEORETICAL ESTIMATION OF ENERGY CONSUMPTION OF WSN ROUTING PROTOCOLS	63
5.5.1	<i>Xiao's Model to Estimate Energy Consumption of WSN Microcontroller</i>	63
5.5.2	<i>Heinzelman's Energy Consumption Estimation on Radio Transceiver</i>	64
 PART 2.		
6.	SECURING BCDCP ROUTING PROTOCOL	67
6.1	SECURING WSN PROTOCOLS VS ENERGY CONSUMPTION	68
6.2	SEBCDCP-MAC INTRODUCTION	69
6.3	SEBCDCP-ENC INTRODUCTION	72
6.4	SECURITY ANALYSIS OF SEBCDCP-MAC AND SEBCDCP-ENC	74
6.4.1	<i>Defending Attacks Breaching Confidentiality</i>	74
6.4.2	<i>Defending Attacks Breaching Integrity</i>	74
6.4.3	<i>Defending Attacks Breaching Authenticity</i>	75
6.4.4	<i>Defending Attacks Breaching Freshness</i>	75
6.5	SECURITY COMPARISON OF BCDCP, SEBCDCP-MAC, AND SEBCDCP-ENC	75
7.	FORMAL VERIFICATION OF THE WIRELESS SENSOR NETWORK ROUTING PROTOCOLS	77
7.1	FORMAL VERIFICATION USING CDVT TOOL	77
7.2	FORMAL VERIFICATION OF SEBCDCP-MAC PROTOCOL	78
7.2.1	<i>SeBCDCP-MAC Initial Assumptions</i>	79
7.2.2	<i>SeBCDCP-MAC Protocol Steps</i>	81
7.2.3	<i>SeBCDCP-MAC Protocol Goals</i>	81

7.2.4	<i>SeBCDCP-MAC Protocol Formal Verification Results</i>	83
7.2.5	<i>Formal Verification of the SeBCDCP-MAC Nonce Updating Steps</i>	83
7.3	FORMAL VERIFICATION OF SeBCDCP-ENC PROTOCOL.....	85
7.3.1	<i>SeBCDCP-ENC Initial Assumptions</i>	86
7.3.2	<i>SeBCDCP-ENC Protocol Steps</i>	87
7.3.3	<i>SeBCDCP-MAC Protocol Goals</i>	87
7.3.4	<i>SeBCDCP-ENC Protocol Formal Verification Results</i>	88
7.3.5	<i>Formal Verification of the SeBCDCP-ENC Nonce Updating Steps</i>	89
8.	THEORETICAL ESTIMATION OF THE ENERGY CONSUMPTION OF WIRELESS SENSOR NETWORK ROUTING PROTOCOLS USING HEINZELMAN’S MODEL	91
8.1	CALCULATING TOTAL NUMBER OF TRANSMIT-RECEIVED BYTES.....	92
8.2	SAMPLE BYTE AMOUNT COMPUTATIONS.....	97
8.3	ESTIMATION PERFORMANCE COMPARISON OF BCDCP, SeBCDCP-MAC, AND SeBCDCP- ENC	100
9.	PERFORMANCE ANALYSIS OF DIFFERENT AES IMPLEMENTATIONS	111
9.1	EMPLOYED SYSTEMS.....	112
9.2	IMPLEMENTATION OF AES ALGORITHMS.....	114
9.3	PERFORMANCE AND POWER CONSUMPTION MEASUREMENT.....	116
9.4	RESULTS ANALYSIS.....	117
9.4.1	<i>Memory usage</i>	117
9.4.2	<i>Execution Time</i>	118
9.4.3	<i>Energy Consumption</i>	120
9.4.4	<i>Which Implementation should be adopted?</i>	122
9.5	COMPARISON OF XIAO’S MODEL ESTIMATION RESULTS AND THE EMPIRICAL MEASUREMENT RESULTS.....	123
10.	IMPLEMENTATION AND PERFORMANCE ANALYSIS OF WIRELESS SENSOR NETWORK ROUTING PROTOCOLS	125
10.1	EMPLOYED SYSTEM.....	126
10.1.1	<i>Differences between the Implemented Protocols and the Original Protocols</i>	127
10.1.2	<i>Packet Structure</i>	128
10.1.3	<i>Implementation Work Scheme</i>	128
10.2	IMPLEMENTING BCDCP.....	131
10.2.1	<i>Implementation’s Event Handler and Work Flow</i>	133

10.3	SEBCDCP-MAC AND SEBCDCP-ENC IMPLEMENTATIONS	136
10.4	EMPIRICAL MEASUREMENT PERFORMANCE COMPARISON AMONG THE IMPLEMENTATIONS...	139
10.5	COMPARISON OF HEINZELMAN’S MODEL ESTIMATION RESULTS AND EMPIRICAL MEASUREMENT RESULTS	141
PART 3.		
11.	REQUIREMENT OF NEW ENERGY CONSUMPTION ESTIMATION MODEL	143
11.1	INADEQUACIES OF XIAO’S MODEL.....	143
11.2	INADEQUACIES OF HEINZELMAN’S MODEL	144
11.3	REQUIREMENT FOR MORE PRECISE WSN ENERGY CONSUMPTION ESTIMATION MODEL....	145
11.4	DETERMINE THE FEASIBILITY OF DESIGNING WSN NODE ENERGY CONSUMPTION ESTIMATION MODEL.....	147
11.4.1	<i>Studying MICAZ’s Hardware Working Characteristics</i>	<i>147</i>
12.	A NOVEL WIRELESS SENSOR NETWORK ROUTING PROTOCOL ENERGY CONSUMPTION ESTIMATION MODEL (PPECEM)	150
12.1	RELATIONSHIP BETWEEN PPECEM AND HEINZELMAN’S AND XIAO’S MODELS	150
12.1.1	<i>Improve Xiao’s Model.....</i>	<i>151</i>
12.1.2	<i>Estimating Energy Consumption on Radio Transceiver</i>	<i>151</i>
12.2	PPECEM’S GENERAL EQUATION	152
12.3	DEDUCING THE PROCESSOR MODULE’S ACTIVE TIME.....	153
12.4	DEDUCING THE WIRELESS COMMUNICATION MODULE’S TRANSMISSION AND RECEPTION ENERGY CONSUMPTION	154
13.	EVALUATE PPECEM	157
13.1	AES-128 ENCRYPTION AND DECRYPTION COMPARISON	157
13.2	BCDCP SETUP PHASE ENERGY CONSUMPTION ESTIMATION	158
13.3	SEBCDCP-ENC SETUP PHASE ENERGY CONSUMPTION COMPUTATION.....	161
13.4	SEBCDCP-MAC SETUP PHASE ENERGY CONSUMPTION COMPUTATION.....	164
13.5	ROUTING PROTOCOLS’ EVALUATION ACCURACY.....	166
14.	CONCLUSION	168
	REFERENCE.....	171

Attachment CD Index

- A. Related Implementations' Source Codes
- B. Used TinyOS Operating System and Development Software
- C. Formal Verification Specification Files For CDVT
- D. Thesis Electronic Format

List of Figures

FIGURE 2.1 GENERAL WIRELESS SENSOR NETWORK ARCHITECTURE	8
FIGURE 2.2 WSN NODE ARCHITECTURE	8
FIGURE 3.1 MULTI-PATH ROUTING	22
FIGURE 3.2 CLUSTER-BASED ROUTING.....	22
FIGURE 3.3 BCDCP PROTOCOL SPECIFICATION	27
FIGURE 3.4 CBKM PROTOCOL SPECIFICATION	31
FIGURE 3.5 SECLEACH PROTOCOL DESCRIPTION.....	33
FIGURE 4.1 SAMPLE OF WORMHOLE ATTACKS.....	42
FIGURE 4.2 SAMPLE OF SYBIL ATTACK.....	42
FIGURE 4.3 SAMPLE OF HELLO FLOOD ATTACK.....	43
FIGURE 4.4 EXAMPLES OF ATTACKS BREACHING INTEGRITY IN BCDCP	46
FIGURE 4.5 EXAMPLES OF ATTACKS BREACHING FRESHNESS IN BCDCP	47
FIGURE 5.1 RC5 ALGORITHM STRUCTURE	53
FIGURE 5.2 AES ENCRYPTION AND DECRYPTION ALGORITHM STRUCTURE.....	55
FIGURE 5.3 MODES OF OPERATIONS	58
FIGURE 5.4 CBC-MAC FLOW CHART.....	61
FIGURE 6.1 SEBCDCP-MAC PROTOCOL SPECIFICATION.....	70
FIGURE 6.2 SEBCDCP-ENC PROTOCOL SPECIFICATION	73
FIGURE 7.1 FORMAL VERIFICATION USING CDVT TOOL.....	78
FIGURE 7.2 SEBCDCP-MAC FORMAL VERIFICATION	83
FIGURE 7.3 SEBCDCP-MAC STEP 2B, 3B FORMAL VERIFICATION RESULTS.....	85
FIGURE 7.4 SEBCDCP-ENC FORMAL VERIFICATION	88
FIGURE 7.5 SEBCDCP-ENC STEP 2B, 3B FORMAL VERIFICATION RESULTS.....	90
FIGURE 8.1 CM RELATIVE MESSAGE OVERHEAD IN DEPENDENCY OF $N_{\text{DATA COM}}$	101
FIGURE 8.2 RATIO OF PROTOCOLS' CH MESSAGE OVERHEAD, IN RELATION WITH $N_{\text{DATA COM}}$	102
FIGURE 8.3 RELATIVE MESSAGE OVERHEAD OF LAST CH IN CH-CH ROUTING PATH, IN DEPENDENCY OF N_{CLUSTER}	104
FIGURE 8.4 RATIO OF THE PROTOCOLS MESSAGE OVERHEAD, IN RELATION WITH THE CLUSTER SIZE ..	105

FIGURE 8.5 RATIO OF THE PROTOCOLS' MESSAGE OVERHEAD, IN RELATION WITH THE SENSING DATA SIZE	107
FIGURE 9.1 ANALYSIS OUTLINE	113
FIGURE 9.2 INTERFACES OF AES IMPLEMENTATIONS.....	114
FIGURE 9.3 OVERALL RELATIVE ENERGY CONSUMPTION.....	122
FIGURE 10.1 A WSN COMPOSED OF MICAZ MOTES AND ONE BASE STATION	126
FIGURE 10.2 WORKFLOW OF THE PROTOCOLS' IMPLEMENTATION IN SETUP PHASE	130
FIGURE 10.3 WORKFLOW OF THE PROTOCOLS' IMPLEMENTATION IN DATA COMMUNICATION PHASE	131
FIGURE 10.4 PACKET FORMATS OF BCDCP IMPLEMENTATION.....	132
FIGURE 10.5 THE RECEIVE EVENT HANDLING FLOW CHART	134
FIGURE 10.6 MAINTIMER TRIGGER EVENT HANDLING FLOW CHART	135
FIGURE 10.7 SeBCDCP-MAC IMPLEMENTATION PACKET FORMAT	137
FIGURE 10.8 SeBCDCP-ENC IMPLEMENTATION PACKET FORMAT	138
FIGURE 11.1 SAMPLE OF MICAZ CURRENT DRAW WAVEFORM (RADIO OFF).....	148
FIGURE 11.2 SAMPLE OF MICAZ CURRENT DRAW WAVEFORM (TRANSMISSION MODE)	149
FIGURE 11.3 SAMPLE OF MICAZ CURRENT DRAW WAVEFORM (RECEPTION OR LISTEN MODE)	149

List of Tables

TABLE 2.1 COMPARISON BETWEEN WSN AND WIRELESS MOBILE AD-HOC NETWORKS.....	12
TABLE 6.1 SECURITY COMPARISON OF THE PROTOCOLS.....	76
TABLE 8.1 NOTATION FOR MESSAGE SIZE COMPUTATIONS.....	92
TABLE 8.2 TOTAL BYTES PER ROUND SUBSCRIPTS.....	93
TABLE 8.3 SAMPLE COMPUTATIONS.....	99
TABLE 9.1 ROM AND RAM USAGE FOR KEY SETUP.....	117
TABLE 9.2 ROM AND RAM USAGE FOR ENCRYPTION.....	118
TABLE 9.3 ROM AND RAM USAGE FOR DECRYPTION.....	118
TABLE 9.4 EXECUTION TIMES FOR KEY SETUP.....	119
TABLE 9.5 EXECUTION TIMES FOR ENCRYPTION.....	119
TABLE 9.6 ENERGY CONSUMPTION OF KEY SETUP.....	120
TABLE 9.7 ENERGY CONSUMPTION OF ENCRYPTION.....	121
TABLE 9.8 ENERGY CONSUMPTION OF DECRYPTION.....	121
TABLE 9.9 COMPUTATIONAL AND REAL WORLD MEASUREMENT RESULTS COMPARISON.....	124
TABLE 10.1 CH SETUP PHASE TIME AND ENERGY MEASUREMENT RESULT.....	140
TABLE 10.2 CM SETUP PHASE TIME AND ENERGY MEASUREMENT RESULT.....	140
TABLE 10.3 CH DATA COMMUNICATION PHASE TIME AND ENERGY MEASUREMENT RESULT.....	140
TABLE 10.4 CM DATA COMMUNICATION PHASE TIME AND ENERGY.....	140
TABLE 10.5 ENERGY CONSUMPTION COMPARISON BETWEEN ESTIMATED AND EMPIRICAL RESULTS ...	141
TABLE 13.1 SOFTWARE-AES ESTIMATION AND EMPIRICAL MEASUREMENT RESULTS COMPARISON	158
TABLE 13.2 PPECEM'S ESTIMATION RESULTS.....	167
TABLE 13.3 RATIO COMPARISON OF THREE PROTOCOLS.....	167

Glossary

Greedy Algorithm: An algorithm that always takes the best immediate, or local, solution while finding an answer. Greedy algorithms find the overall, or globally, optimal solution for some optimization problems, but may find less-than-optimal solutions for some instances of other problems [Bla05].

Microcontroller: A simple computer on an integrated circuit. Microcontrollers have minimum components like processor core, RAM, and programmable input and output [Hea03].

Nonce: Nonce is an arbitrary number. Nonce is used only once within a session and usually is generated randomly or pseudo-randomly. Nonce is used widely in security protocols to ensure old communications will not be reused [Rog04].

P2P: PEER-TO-PEER (abbreviated to P2P) systems and applications are distributed systems without any centralized control or hierarchical organization, in which each node runs software with equivalent functionality [SML03].

TDMA: Time division multiple access (TDMA) is a channel access method for shared medium networks. In a TDMA cellular radio system, several users time-share a common carrier frequency to communicate with the base station. Each user, transmitting low bit-rate digitized speech or other digital data, is allocated one or more timeslots within a frame in the downstream (base to users) and upstream (users to base) directions [FAG95].

Part 1.

Background

1. Introduction

1.1 General Introduction

Wireless Sensor Networks (WSN) are ad hoc networks composed of spatially distributed small autonomous network sensor nodes, called Wireless Sensor Nodes (WSN Nodes), to cooperatively monitor physical or environmental conditions [RMG02]. WSN nodes are loaded with one or multiple sensors to collect information about the surrounding physical conditions, such as a patient's body condition, or environmental data, i.e. temperature, humidity, light, etc., and transfer this to the Base Station (BS). The BS is responsible for gathering the information aggregated by the WSN nodes, and managing communications with the outside world, e.g. processing the gathered sensing data and displaying it to the end users and managing the network topology or generating session keys for security protection [RoM04].

WSNs have the same characteristics as traditional wireless communication and networking. Additional characteristics of WSN are small scale, low cost, sometime irreplaceable power supply, multi-functioning microcontroller, and limited memory space [AlK04]. Not only can the WSN nodes be deployed in a large area (over 100KM), the deployment density can be high. The multi-functioning microcontroller provides strong processing capability, enabling

relatively more complex tasks like data fusion, intelligence clustering and routing, and some security protections such as encryption/decryption and Message Authentication Code (MAC) [RoM04] to be carried on.

Designing a good routing protocol for WSN is a challenging task as the WSN routing protocol designers have to cover inherited design difficulties from wireless ad hoc networks [KaW02]; moreover, the large amount of WSN nodes, the unattended WSN node operations, and most importantly, the restrained power source all increase the design difficulties [AlK04]. Throughout this chapter some fundamental WSN routing protocols' vulnerabilities and the requirement for comprehensive energy consumption estimation models are introduced.

1.2 Necessity of Security Protections to WSN Routing Protocols and Its Price

Along with the rapid development of WSN systems, various attacks against the security of WSN routing protocols have arisen [HPJ02] [KaW02] [KGD08] [LHJ05] [NLD08] [RMB09] [SJS10] [Uou02]. As a result the requirement for more security protection has been increased. Routing protocols' security has become one of the major concerns of WSN [PHC06]. WSN routing protocols should be secured once they are involved in sensitive data transmission [RoM04] [PHC06] [PSW04] [SKS11] [ShP04]. The messages exchanging among the WSN nodes and the BS in a WSN routing protocol should be secured to protect the messages' confidentiality, integrity, freshness, and authenticity. Yet, the WSN node, with hardware security support [Chi04] or not, requires extra time and energy for security computations (such as message encryption and decryption or MAC generation [LDH06]), and includes the energy consumed by the radio transceiver for the extended message overhead [LKS10]. This added security overhead

violates the energy conservation objective of WSN design [ASS02]. Therefore, although numerous energy-efficient, robust, and reliable WSN routing protocols for Wireless Sensor Networks have been announced [OFV07] [MMB05] [HCB00] [SRS07] [Zha10] [YBH10] [You04] [MRK05], many of them have no security protection [MMB05] [HCB00] [Zha10] [YBH10] [You04] [MRK05].

1.3 Requirements of Energy Consumption Estimation

As already mentioned, the WSN routing protocol should be secured once it is involved in sensitive data transmission. On the necessity of securing the routing protocols, there is inevitable energy cost for protecting the protocol. To choose a secured WSN routing protocol for their network, the WSN architects may ask: “which WSN secured routing protocol should I adopt?” or “is the adopted routing protocol energy efficient?” Thus, both the WSN routing protocol designers and WSN architects need to effectively and accurately evaluate the energy consumption performance of the routing protocols. With the comparable evaluation results, the WSN routing protocol designers can further validate the energy efficiency of their routing protocols over the competing protocols, and the WSN architects can decide the price of adding security to their protocols or further estimate the network lifetime of their WSNs.

To estimate the energy consumption of a WSN routing protocol, one way is to implement the protocol in an experimental WSN and measure the energy consumption on the WSN nodes. This method can validate both the protocol’s applicability to the target WSN hardware and the energy efficiency.

The other approach is to theoretically estimate the energy consumption using mathematical models. The theoretical estimation models are popular methods to

estimate the WSN routing protocols' energy consumption level. Heinzelman et al. [HCB02] announced a general radio transceiver energy consumption model. The model takes the transmission distance as one of the key factors, and has been widely used in many papers to evaluate their announced WSN routing protocols' energy consumption performance [OFV07] [MMB05] [You04] [MRK05].

Nowadays, more and more WSN routing protocols are protected by security methods like encryption or MAC. Xiao et al found a way to simulate the actual processor overhead (in μs) while encrypting and decrypting a block of data using AES algorithm [XCS06]. In their research, they analyzed the algorithm of AES, and deduced the minimum number of processing cycles required for performing an AES encryption and decryption.

Both Heinzelman et al and Xiao et al's models have advantages and disadvantages, but neither of them is comprehensive enough to evaluate WSN routing protocols' energy consumption. Heinzelman's model overlooked the power consumed by the microcontroller; Xiao's model, on the other hand, does not involve the energy consumption on the radio transceiver's side. Neglecting power consumption on either microcontroller or radio transceiver may lead to inadequate or incorrect computational results. Thus, a more comprehensive WSN routing protocol energy consumption estimation model is required.

1.4 Objective of the Thesis

The main objective of this research is to identify and estimate the energy consumption of routing protocols for WSN. A study of WSN routing protocols, both unsecured and secured, will be undertaken to evaluate WSN characteristics and limitations as well as attacks against WSN and their countermeasures. A

new secure and efficient routing protocol will be proposed and formally verified. The cost of security in terms of energy consumption of this secure protocol will be established using energy consumption estimation models. Also, the secure protocol will be implemented and energy consumption measurements will be carried out. Comparison of the estimated values against the measurements will determine the precision of the estimation models.

Another objective is to provide an improved energy consumption estimation model for WSN routing protocols. This improved energy consumption model should effectively and precisely evaluate protocols' power dissipation.

1.5 Structure of the Thesis

The thesis is divided into three parts. The first part includes five chapters: the first chapter gives a short introduction to WSN, their security vulnerabilities, and studies why a comprehensive energy consumption estimation model is required. Chapter two discusses the WSN definitions, characteristics, and limitations. Chapter three describes the review of existing WSN routing protocols. In the fourth chapter, the attacks against WSN are presented in detail. The fifth chapter concerns the countermeasures to the attacks listed in chapter four, and analyzes the security of two WSN routing protocols.

The second part involves five chapters: in chapter six, a new secure WSN routing protocol is proposed in two parts. The formal security verification of the secured WSN routing protocols is described in Chapter seven. In Chapter eight, the energy consumption estimation of three WSN routing protocols are detailed. In the ninth and tenth chapters, the implementations of symmetric encryption algorithm and three routing protocols on an experimental WSN, and the

empirical energy consumption measurement results, along with their performance comparison against the traditional models' calculation results are presented.

The third part contains four chapters. The analysis of the traditional mathematical models is discussed in chapter eleven, together with the feasibility of a more comprehensive energy consumption estimation model for WSN routing protocols. In chapter twelve, a new comprehensive WSN routing protocol energy consumption estimation model is presented. The proposed model is then evaluated by comparing its computed results to several routing protocols against implementations' measurement results. The computation procedures and comparison results are presented in Chapter thirteen. Chapter fourteen is the conclusion.

2. Wireless Sensor Networks

2.1 Wireless Sensor Network Definition

Wireless Sensor Networks (WSN), which have been made popular by UC Berkeley [LMP05], are area wireless networks consisting of spatially distributed autonomous devices, called WSN nodes, and using sensors to cooperatively monitor physical or environmental conditions (such as temperature, sound, vibration, pressure, motion or pollutants) at different locations [RoM04]. Originally, WSNs were motivated to be developed as military applications [ASS02]. Nowadays it is used in many civilian application areas including environmental and habitat monitoring, healthcare applications, home automation, and traffic control [RMG02].

Figure 2.1 describes a general WSN architecture. The randomly deployed WSN nodes send their sensed data to the Base Station (BS) via the routing nodes. The BS acts mainly as the gateway between the WSN and the outside world. The Routing nodes and the Sensor nodes in the same network can be homogeneous or heterogeneous. Based on WSN's design, the role of router node can be either fixed on some WSN nodes, or shifting among WSN nodes in the network.

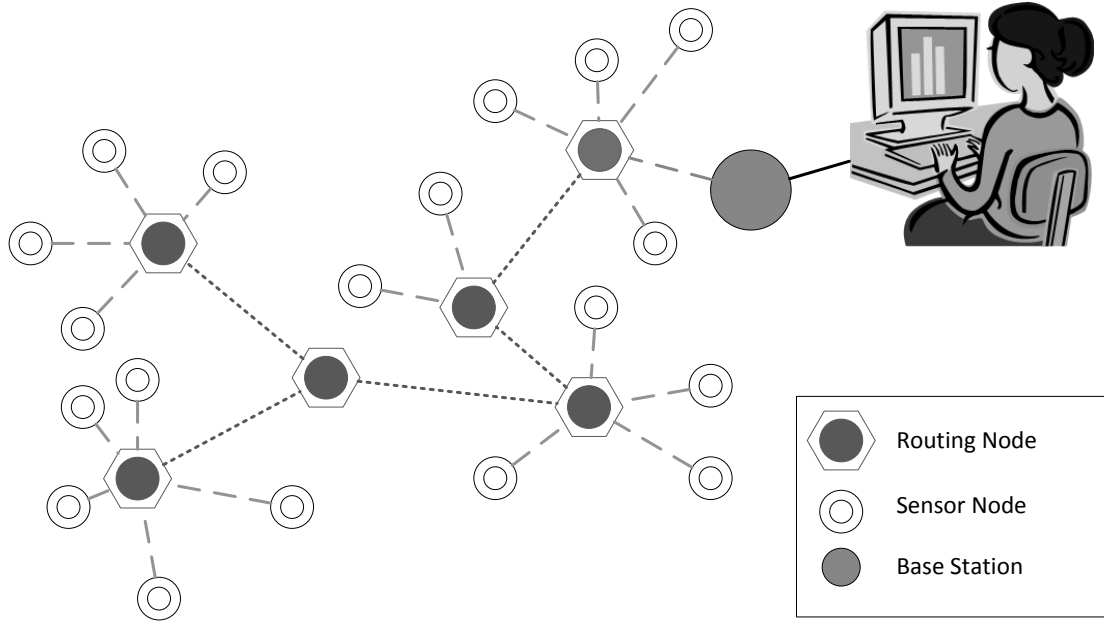


Figure 2.1 General Wireless Sensor Network Architecture

2.2 Wireless Sensor Node

A common Wireless Sensor Node (WSN node) is composed of at least one sensor module, one processor module, one wireless communication module, and one power supply module [ASS02], as shown in Figure 2.2.

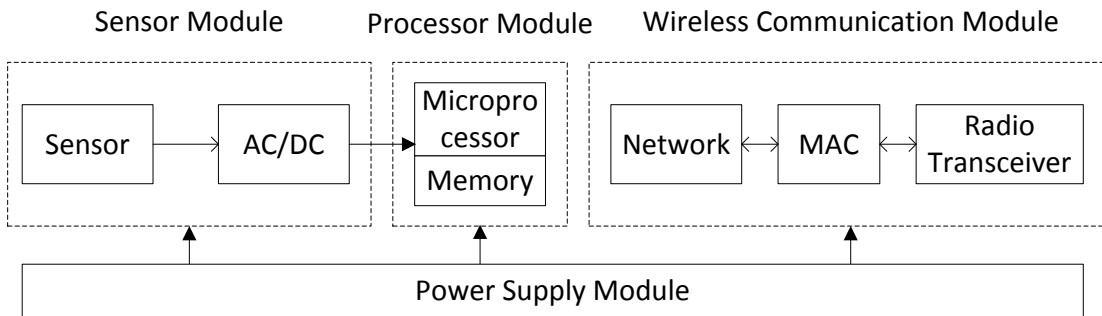


Figure 2.2 WSN Node Architecture

Normally, the sensor module collects the observed surrounding environmental analog information such as light, sound, shocks, etc, converts it to the digital signal via the analog to digital converter (ADC), and then transfers to the

processor unit. The processor module manages the cooperation between the units in the sensor, the collaborations between other WSN nodes in the network, and may perform security computations such as data encryption and MAC digest generation. The communication between other WSN nodes and the BS is done by the wireless communication module. All these three units are powered by the power unit, which is usually composed by battery [RoM04].

2.3 The Characteristics of Wireless Sensor Network

In general, WSN have the characteristics of Large Scale, self-organizing, dynamic, application-oriented, and data-centric.

2.3.1 Large Scale Network

The term “Large Scale Network” in WSN either indicates that the WSN nodes are deployed in a geographically large area, such as forest fireproofing and environmental monitoring applications; or means the WSN nodes are densely deployed in a relatively small area, to achieve precise sensing tasks, i.e., the mobile object tracking system [SEF10].

The large scale deployment of WSN has the following advantages: the sensing results of one monitoring area from different sensing angles helps obtain better information Signal-to-Noise-Ratio (SNR); the distributed sensed data processing helps increase the overall monitoring result precision and lower the precision requirement to single WSN node; the redundancy of the deployed WSN nodes improves the fault-tolerance of the network; and a large amount of WSN nodes helps increase the coverage of the sensing area, and decrease the blind zone.

2.3.2 Self-organized Network

The WSN nodes are often deployed randomly in the monitoring area with no initial infrastructure. Meanwhile, the WSN nodes in the network are not aware of their own geographic locations or of the presence of any neighbors. This requires the WSN nodes to be capable of self-organizing, self-configuring, and self-management. The nodes organize themselves to a network system via the execution of the pre-programmed topology control mechanism and network protocol implementation.

2.3.3 Dynamic Network

WSN's network topology may change over time, and the differences in relation to a traditional wired network, are due to, but not limited by, the following causes:

- Unpredictable factors such as thunder strike, adversary destroying the WSN nodes, or the WSN nodes' battery being drained, which may cause the WSN nodes' to malfunction or die;
- The environmental conditions' change, like heavy rain or deep fog, that would influence the WSN nodes' communication bandwidth, distance, or even intermittency;
- Movement of the WSN nodes or monitoring objectives;
- Joining of new nodes.

The WSN nodes should be capable of adapting the changes, and can re-construct the new network topology dynamically.

2.3.4 Application Oriented Network

Different WSN applications have different sensing requirements. These differences necessitate diverse hardware and software platforms, network topologies, and security requirements. The extreme energy efficiency requirements force tight cooperation among hardware, software, and network protocols, to serve one specific application completely. Designing a unified communication protocol is infeasible for WSN applications.

2.3.5 Data Centric Network

Unlike the traditional Internet's address centric characteristic [Bar64], WSN is more like a data centric network.

In WSN, as the network topology is dynamic and the end user of the network focuses only on the collected data, a single WSN node is not as important as the PC connected to Internet. When the user wants to acquire certain data in the network, he or she can make an inquiry to the entire network. WSN nodes in the network collaborate to finish the task and transmit the data to the BS, and then to the users. The user does not have to know which WSN nodes collected the data or how it is transmitted. All he or she cares about is the data. The data centric idea closes the natural language communication custom.

2.3.6 WSN and Mobile Ad-hoc Networks Characteristics Comparison

Wireless mobile ad-hoc networks are wireless multiple-hop mobile networks grouped by hundreds to thousands of mobile nodes, and are supposed to have

2. Wireless Sensor Networks

unlimited power supply [Mis10] [Oth08]. The main purpose of the network is to transmit a high quality multimedia information stream via dynamic routing and mobile management technologies [KrB04].

Table 2.1 Comparison between WSN and Wireless Mobile ad-hoc Networks

	Wireless Mobile ad-hoc Networks	WSN
Objective	Data exchange	Environment monitor, wireless control, and wireless communication
Device	Laptops, Tablets, Mobile phones	Tiny Node with weak CPU and storage
Routing	All-to-all routing	P-to-P communicate with router node, or broadcast
Mobility	Yes	Usually no mobility
Structure	Less frequently change	Frequently change
Scale	hundreds	Thousands or more
Battery	Unlimited - rechargeable	Limited

WSN and mobile ad-hoc networks share some characteristics in common, such as wireless communication. However, they are quite different in some ways [ASS02] (Cf. Table 2.1). Firstly, different to ad hoc networks' one-fold objective of exchanging data, WSN are integrated with environmental monitoring, wireless control, and wireless communication network systems. Secondly, WSN are composed of tiny WSN nodes with relatively weak processor (8MHz) and limited storage space, while Wireless mobile ad-hoc networks are often organized by more powerful devices like laptops and/or mobile phones. Second, the routing strategy of WSN is P2P communication between sensor nodes and router nodes, or broadcasting, but Wireless ad-hoc networks adopt all-to-all routing: each node can communicate with any other nodes in the network. Thirdly, WSN nodes are usually fixed, while ad-hoc network devices are mobilized. Fourthly, due to the out-door deployment character, WSN nodes are more easily damaged or broken, which hence makes the changing of the routing structure more frequent than on ad-hoc networks. Fifthly, the amount of nodes in a WSN can be far more than that of ad-hoc networks, and have denser

deployment. And lastly, the nodes in the wireless mobile ad-hoc network are considered as unlimited power supply, while the WSN node's battery is limited.

2.4 Limitations of WSN Nodes

WSN Nodes have a number of limitations to restrict their capabilities. The limitations include power source, communication range, plus the relatively weak microcontroller and small volume storage space [RoM04].

2.4.1 Limited Power Supply

WSN nodes are small-scale, low-cost, and low power consumption embedded devices, which decide the power supply module is small-sized with limited power storage.

2.4.2 Limited Communication Capability

Due to the large power consumption required for transmitting and receiving message packets, the wireless communication module consumes most energy among the three modules.

The relationship between the received signal power and the transmission distance has been well established (Eq. 2.1) [PaL05]:

$$P_r = \frac{P_0}{d^\alpha} \quad (\text{Eq. 2.1})$$

While P_0 is the transmission signal power and P_r is the received signal power. d is the distance between the sender and receptor, and α is the distance-power

gradient ($\alpha \geq 2$). Thus, the longer transmission distance, the more power the radio transceiver has to consume. Thus, the wireless communication module cannot use too much energy to send a message to a distant range. Normally 100 meters is the maximum transmission range [SiC01].

2.4.3 Limited Computation and Storage Capacity

WSN nodes are small-scale, low-cost, and low power consumption embedded devices. These limitations decide the relatively weak microcontroller computation capability and small-volume storage capacity. A typical WSN node should be capable of completing the following tasks: aggregating, processing, and managing the sensed data; answering the sink node or base station's requests; controlling it based on the commands sent by the sink node / base station, etc. How to utilize the limited computation and storage resources of the sensor hardware, in order to complete the cooperative tasks, is one of the main challenges in designing WSNs.

2.5 Energy Consumption of Wireless Sensor Networks

Among the units in a WSN node, the processor module and the wireless communication module are two main energy consumers in a WSN node [Kav10]. Among the two modules, the wireless communication module consumes more energy than the processor module as transmitting and receiving the message packets over the air requires a large amount of energy.

2.5.1 The Importance of Energy Conservation

Except for unexpected causes such as hardware malfunctioning or malicious destruction, the most common reason for causing a WSN node's death is battery draining. The death of a few nodes may cause changes to the network's topology or reorganization. As the dead node count increases, more nodes may be isolated from the network due to the absence of the router node within their communication range. As this process continues, the network eventually dies. In this regard, WSN's lifetime has strong dependence on the amount of living WSN nodes. Energy conservation has become one of the most important design objectives of WSN.

Energy Conservation on Processor Module

Currently, along with the advance of low power dissipation circuit and system design techniques, many ultra-low power consumption microprocessors have been developed. Apart from the lowering of the power dissipation capability, modern microprocessors are currently supporting dynamic power management (DPM) [PrS11] and dynamic voltage scaling (DVS) [HGS11]. DPM is the processor's capacity of shutting down or setting to sleep the mode of certain modules when there is no task for them to complete. DVS mainly deals with lowering the processor's working voltage to save power.

Energy Conservation on Wireless Communication Module

For energy saving purposes, a wireless communication module generally has three statuses: transmission, reception (or listening), and sleep (or idle) [BBV07]. Each status has different energy consumption. In general, transmission status requires maximized energy for sending the message bytes some distance. Sleep status requires a minimum or no power as the radio transceiver in this status is

shut down or is put in “deep sleep” mode. Listening and reception status have same energy requirements as either of them needs to put the radio transceiver in “listen” mode to monitor or process the incoming messages [SiC01].

Energy Conservation on Routing Protocol Design

The WSN’s routing has a strong impact on the network lifetime [Fot10]. Thus, energy conservation has become one of the main objectives of designing a secured routing protocol. For conserving energy consumption, the design ideas of WSN routing protocols are:

- Minimizing the message payload
- Minimizing the number of messages
- Minimizing the radio transceiver’s active time
- Minimizing the WSN node’s microcontroller computation time.

After a WSN routing protocol is designed, it is instructive to estimate its energy consumption level, since energy efficiency is paramount for WSN.

2.6 Application Domains of Wireless Sensor Networks

WSN can be applied in many application domains including military, commercial, environmental, and healthcare domains.

2.6.1 Military Application Area

WSN was first motivated by military requirements. The rapid deployment, self-organization, and fault tolerance characteristics of WSN led to them being chosen as sensing technology for military C4ISRT (command, control, communications,

computing, intelligence, surveillance, reconnaissance, and targeting) system [Slo05]. The low-cost and dense deployment characteristics make long system lifetime possible in a hostile environment – the enemies' destruction of a certain amount of WSN nodes does not cause as much harm as a traditional wired sensor network. The military applications are in the following categories:

Friendly force information interaction: the knowledge of friendly forces in the battlefield, including the geographical location, the equipment, and ammunition status, offers many advantages, such as avoiding friendly fire and efficient commanding. This can be done by using WSN technology. Every troop, piece of equipment, and critical ammunition is equipped with a WSN node to report on the status regularly. The data is forwarded via the router nodes to the base.

Battlefield surveillance: the sensors can be deployed, concealed and rapidly sent to the surveillance area, to watch the activities of the opposition forces. The deployment areas can be critical terrains, approach routes, paths, or straits.

Reconnaissance of opposing forces and terrain: Sensor networks can be deployed in critical terrains, and some timely intelligence about the opposing forces and terrain can be gathered within minutes before the opposing forces can intercept them.

Examples of WSN Military Applications:

VigilNet [HKL06] was designed for completing typical ground surveillance tasks: to detect, alert, and track the occurrence of events of interest in hostile regions. The system was designed to achieve longevity (a few months), adjustable sensitivity (for complying with different missions), stealthiness (zero

communication exposure in the absence of significant events), and effectiveness (accurate and hardly noticeable in hostile area). The architecture was based on TinyOS [LMP05], which provides essential support for running the system.

Counter-sniper [LNR05]: An ad hoc wireless sensor network system for detecting snipers was presented. The system was able to detect shockwave and muzzle blast to estimate the bullets' time of arrival and locate the sniper with the precision of 20-30 cm once his shooting is detected within the network coverage.

2.6.2 Environmental Application Area

Environmental applications are significant drivers for WSN development [Atm03]. In the near future, WSN will play a key role in sensing, collecting, and disseminating information in environmental applications [TAH02]. It enables scientists to measure properties that have not previously been observable.

The applications include:

- Tracking the movements of animals.
- Deploying nodes to the forest to detect fire location.
- Monitoring environment conditions.
- Chemical detections, flood detection, air and soil pollution detection.

Examples of WSN Environmental Applications:

Light Under Shrub Thicket for Environmental Research (LUSTER) [SWC07] was a system designed of EWSNs (Environmental WSN), for monitoring the effects of sunlight, along with temperature, humidity, CO₂, and soil moisture.

The main characteristics of the system were hierarchical architecture, distributed reliable storage, deployment time validation capability, and delay-tolerance networking. The system also achieved reliable storage by adopting distributed storage nodes to collect and store data sent from WSN nodes, waiting for further online querying or manual collection.

Habitat Monitoring, presented in [SMP04], concerned large scale habitat monitoring. The main objective of the application was to direct human observation and monitor average temperature at midnight in the burrows and on the surface. The system adopted two topologies: single hop and multiple hops via routing tree.

2.6.3 Health Application Area

WSN' health applications can offer great support in medical areas, such as when deploying a WSN inside a house and on the human body to:

- Remotely monitor health data on human body
- Warn patient to take pills
- Monitor the habitat.

Example of WSN Health Applications:

A **Wearable Wireless Body/Personal Area Network (WWBAN)** prototype was presented in [MOJ06]. The basic architecture included wireless medical WSN nodes, which were responsible for monitoring muscle activity, blood pressure, respiration, motion, and brain electrical activities. A personal server (PS) offers a transparent interface between WSN nodes, users, and remote medical servers.

The communication between wireless medical WSN nodes and the PS was via short-range wireless network where the PS is connected to a medical server accessed via the Internet.

2.6.4 Commercial Application Area

Commercial applications of WSN range from personal in-house automation and industrial hardware assembly, to office air conditioning control and transportation vehicle speed and location tracking.

Example of WSN Commercial Applications:

An application of commercial WSN was **Wisden**, a Structural Health Monitoring System to detect and localize damage in buildings, bridges, ships, and aircraft [XRC04]. The goal of the project was to generate the abstraction of a data acquisition system, providing vibration data of the structure where tens of WSN nodes were deployed. The WSN nodes could sense up to three channels' vibration data. A base station logged data transmitted from the nodes. Nodes self-configure to form a tree-topology.

3. Review of Wireless Sensor Network Routing Protocols

Two main approaches currently exist for WSN routing protocols' development: multi-path routing and cluster-based routing. In this chapter, a number of published energy efficient WSN routing protocols is discussed.

3.1 Multi-path Routing and Cluster-based Routing

Routing is an essential part of any WSN system [RoM04]. Various routing strategies have been developed, each with its own virtues and limitations. Currently, there are two main approaches to routing in WSN: multi-path routing and cluster-based routing [Sin10]. Multi-path routing protocols, such as [LDZ10], and [Zha10], provide alternative routing paths when the main routing path has become unavailable (cf. Figure 3.1). However, due to unbalanced energy consumption, WSN nodes on busy routing paths may drain their batteries faster than other nodes. Thus, the overall network's lifetime is shortened. E.g., in Figure 3.1, the node C and F would die faster than other WSN nodes as they are located in the busiest routing path ($A \rightarrow C \rightarrow F \rightarrow$ Base Station (BS)).

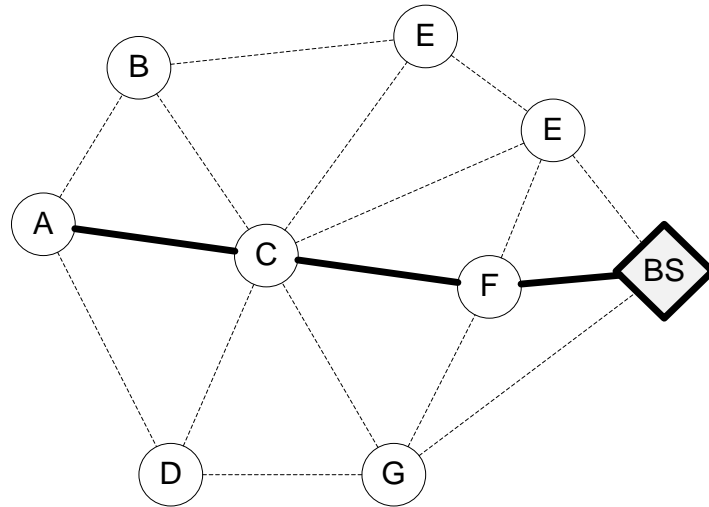


Figure 3.1 Multi-path Routing

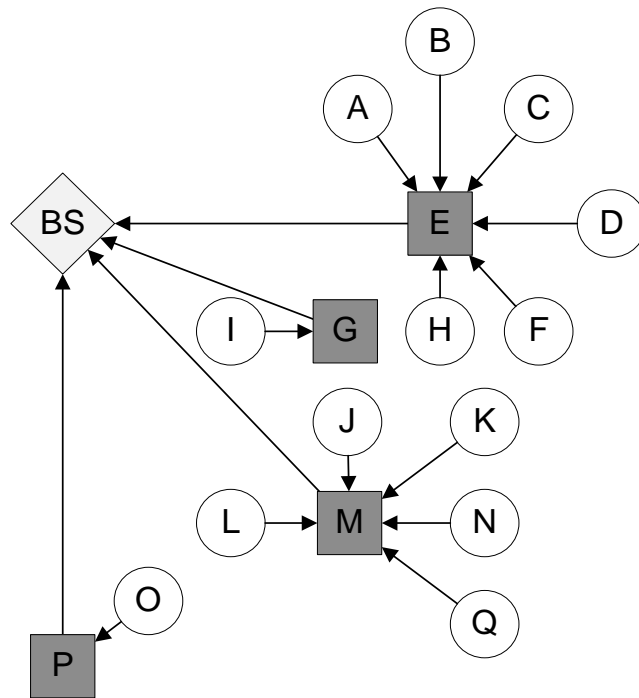


Figure 3.2 Cluster-based Routing

On the other hand, in cluster-based routing protocols, such as [HCB00], [MMB05], [SRS07], and [YBH10], the WSN is divided into multiple clusters, each with one or more cluster heads (CHs). CHs are located within single-hop communication

distance away from their cluster members (CMs). CHs are responsible for forwarding messages between their CMs and the BS. The role of CH is rotated among the CMs, to distribute the additional power consumption caused by routing duties equally among the nodes in a network (Cf. Figure 3.2).

3.2 Low Energy Adaptive Clustering Hierarchy (LEACH)

LEACH is a self-organizing, adaptive clustering protocol that uses randomization to distribute the energy load evenly among the sensors in the network [HCB00]. In LEACH, the WSN nodes organize clusters locally. In each cluster, one node acts as the CH to aggregate the data from all its CMs. The aggregation results are then sent to the BS. To maximize the channel capacity usage and avoid congestion, Time division multiple access (TDMA) scheduling [FAG95] is used for CH regulating the different time slots for each CM to send their sensed data.

One of the disadvantages of fixed WSN nodes being selected as routing nodes or CHs through the system lifetime was the fast battery drain. The dead router nodes or CHs will influence the network topology and leave “blind” monitoring areas. Furthermore, the increasing number of dead WSN nodes starts to isolate a number of WSN nodes from the network, and drastically shorten the network lifetime [RMG02].

In order to solve the above problem, the authors of [HCB00] suggested two strategies: utilizing the cluster network topology and randomly rotating the CHs. In their solution, the network is divided into a number of clusters. Each cluster contains one CH. The simplified CM → CH → BS routing path helps reduce the isolated WSN nodes. The randomized high-energy-consumption CH role

rotation strategy helps average the energy burden to every node in the network. In addition, the CHs in LEACH perform data fusion to “compress” the message bytes sent to the BS, which further reduces the energy consumption.

In LEACH, once a WSN node has not been CH for G rounds, where G is a pre-defined number, it generates a pseudo-random number between 0 and 1. If the generated random number is less than the threshold T , it elects itself as a potential CH. The threshold T varies along with the number of rounds elapsed. Once the WSN node is willing to be the CH for the current round, it broadcasts this willingness to the network, using predefined initial signal strength. The WSN nodes who receive this will compare its signal strength against the other messages, and select the sender of the message with the strongest signal strength to be their CHs, as the stronger signal strength means a shorter distance between the CM and the CH. The CMs then transmit an acknowledgement message back to the CHs. After collecting all acknowledgements, the CHs will generate a sensed data TDMA scheduling for its CMs, and broadcast it to them.

In the data communication phase, the CMs who received the time scheduling message will wait for their own time slot to send their sensed data. The CH must keep its radio transceiver on during the whole data packet receiving procedure. When the CH has received all data messages from its members, it performs a data fusion procedure to compress the messages, and sends them to the BS. The data transmission phase will iterate a fixed number of times, until the current round is finished.

3.3 Power-Efficient Gathering in Sensor Information Systems (PEGASIS)

Although the CMs in LEACH have the option of deciding the CH based on their messages' signal strength (the stronger signal means the closer the distance), the random rotation of the CH role makes the distance to the CH uncertain and it may be too far. Once in a large network, if two WSN nodes are not within the communication range of each other, a multi-hop strategy may have to be performed, which LEACH does not provide for in any aspect. PEGASIS [LRS02] was designed to improve this situation.

The main idea in PEGASIS is for each WSN node to receive from and transmit to close neighbors, and to allow each one to alternatively lead the transmission to the BS. PEGASIS assumes all WSN nodes in the network are aware of their geographic locations. They also know the geographic locations of their neighbors by broadcasting and receiving HELLO messages containing senders' location information. The WSN nodes group to a chain based on a greedy algorithm. The chain grouping can be done by either the BS or the WSN nodes locally, and broadcasts to the network. Either way, the chain organizer node needs to obtain the neighboring nodes' geographic locations.

The WSN nodes with furthest distance to the BS will be the first node in the chain. The closest neighbor to this node will be the next node on the chain. The chain keeps expanding in this manner to form the greed chain. To transmit data, the node receives the data from one neighbor, fuses its own, and transmits to the next neighbor in the chain. In each round, a WSN node i in the chain exists to send the final fusion data to the BS. Node i is called the *leader* node. The BS will determine the leader randomly but ensure that a leader node will not be re-elected for a predefined number of rounds

In a given round, the idea of a control token is adopted to avoid the conflict of two or more WSN nodes in a chain trying to transmit the data message at the same time. The WSN node that possesses the control token can transmit the data along with the token to its neighbor.

3.4 Base-Station Controlled Dynamic Clustering Protocol (BCDCP)

The Base-Station Controlled Dynamic Clustering protocol (BCDCP) [MMB05] uses a cluster routing approach to evenly distribute the energy dissipation among all WSN nodes in the network. In BCDCP, the geographic monitoring area is manually defined into blocks prior to the WSN node deployment. Each geographic block is assigned an individual location ID. Pre-installing the ID of the designated geographic block into the WSN nodes makes them aware of their physical locations. BCDCP uses the following network model:

- A fixed BS is located far away from the WSN nodes.
- The WSN nodes are energy constrained with a uniform initial energy allocation.
- The nodes are equipped with power control capabilities to vary their transmission power.
- Each node senses the environment at fixed time intervals and always has data to send to the BS.
- All WSN nodes are immobile.
- The WSN nodes are capable of operating in two modes: CH mode and CM mode.

The BCDCP protocol works in rounds, where each round consists of a setup phase followed by a fixed number of data communication phases (cf. Figure 3.3).

3. Review of Wireless Sensor Network Routing Protocols

In the setup phase, each WSN nodes sends its location ID, along with its remaining energy level, to the BS (1st step of Figure 3.3). Based on the received location and remaining energy information, the BS divides the WSN nodes of the network into clusters for the current round. In each cluster, the BS assigns to one node the role of CH. BS also creates TDMA schedule and regulates the time slots for each member to send their sensed data to the CHs. BS then sends every CH a CH-to-CH routing path, its CM ID set, and the time slots for each CM (step 2 of Figure 3.3). For every CM, the BS sends it its CH's ID, and the time slot for them to send data to the CH (step 3 of Figure 3.3).

Setup Phase	
1. $\delta \rightarrow BS$:	$ID_{\delta}, Loc_{\delta}, RE$
2. $BS \rightarrow C$:	$ID_C, \langle Path \rangle, \langle MID \rangle, \langle TSN \rangle$
3. $BS \rightarrow A$:	ID_A, ID_C, TS
Data Communication Phase	
1. $A \rightarrow C$:	$ID_A, Data$
2. $C \rightarrow BS$:	$ID_C, Fdata$

Legend:

- δ : Every WSN node in the network
- A: Arbitrary WSN node
- C: Cluster head
- BS: Base station
- ID_X : ID of WSN node X
- Loc_X : Location information of node X
- RE: Remaining energy
- $\langle Path \rangle$: A sequence of cluster head IDs indicating the cluster head-to-cluster head routing path
- $\langle MID \rangle$: The set of member node IDs of a cluster
- TS: Time schedule for a single WSN node
- $\langle TSN \rangle$: The set of time schedules for nodes of a cluster
- Data: The sensed data
- Fdata: The fusion data.

Figure 3.3 BCDCP Protocol Specification

In the data communication phase, each WSN node transmits its sensing data to its CH at the designated time slot (Figure 3.3's step 4). Once the CH has received data from all its CMs, it sends the aggregated data to the BS via the CH-to-CH routing (Figure 3.3's step 5). After a fixed number of data communication phases the setup phase is re-entered and the network is re-configured.

3.5 Cluster Based Key Management Protocol (CBKM)

A Cluster Based Key Management protocol (CBKM) for WSN was presented in [SFQ08]. The protocol is designed to be a securely dynamic clustering protocol with a flexible key management capability. The protocol utilizes the Exclusion Basis System (EBS) algorithm [KhS06] to manage group encryption keys and the LEACH protocol [HCB00] to elect new nodes as CHs. The security protection used in CBKM, such as encryption and key update, will be introduced in Chapter 5.

The CBKM protocol is based on the following assumptions:

- Each WSN node possesses a unique ID and a secret key initially (node key). The node key is known only by it and the BS.
- Each cluster has a unique cluster ID X and cluster key K_{CX} . K_{CX} is known by every CM and the CH in the cluster. CMs can only communicate with each other through the CH.
- Each WSN node possesses a global initiate key K_{init} prior to deployment, which is used for the first cluster generating process. K_{init} is discarded once the first cluster setup is completed.
- CHs are ordinary WSN nodes with the responsibility of integrating their members' data together and forwarding them to the BS.

- The BS has more power and can communicate with all CHs in the network with a single hop. The base station processes all sensor data and is in charge of managing keys for the network. In addition, the base station knows the geographic position of all WSN nodes in the network.
- Prior to deployment, BS generates a pool of keys, each having an individual key ID. Each node is randomly assigned a number of keys out of the key pool (called sub-key-pool). BS tries to find a set of nodes that any two of them can find a shared key within their sub-key-pool. If one or more nodes are captured by the adversary, new keys are distributed using the only keys that are not known to the captured nodes. This is called EBS key management.

The CBKM protocol is composed of two sub-protocols: Firstly, the initial clustering protocol, in which the nodes of the WSN are organized into clusters and the CHs are elected; secondly, the Cluster Role Changing Protocol, which re-elects new CHs to allow nodes to change roles to save energy.

The initial clustering protocol (as shown in Figure 3.4) executes immediately after system initialisation. Some WSN nodes such as node I self-elect to be the CHs, based on the algorithm presented in LEACH. These nodes broadcast a HELLO message that includes their identity. This message is encrypted with the key K_{INIT} . The WSN nodes (A) in the broadcasting range of the elected CHs receive (potentially multiple) HELLO messages (*dataHELLO*) and decide to which cluster they will belong. Node A sends an acknowledgement message (*dataACK*), consisting of A's identity and the ACK flag encrypted with K_{INIT} , to the selected cluster's head (I). Each CH will wait for a defined period of time to receive acknowledgement messages. Afterwards, the CHs send a message to the BS informing it about the cluster configuration. This messages contains the IDs of all nodes in the cluster and is encrypted with the key which I shares with BS

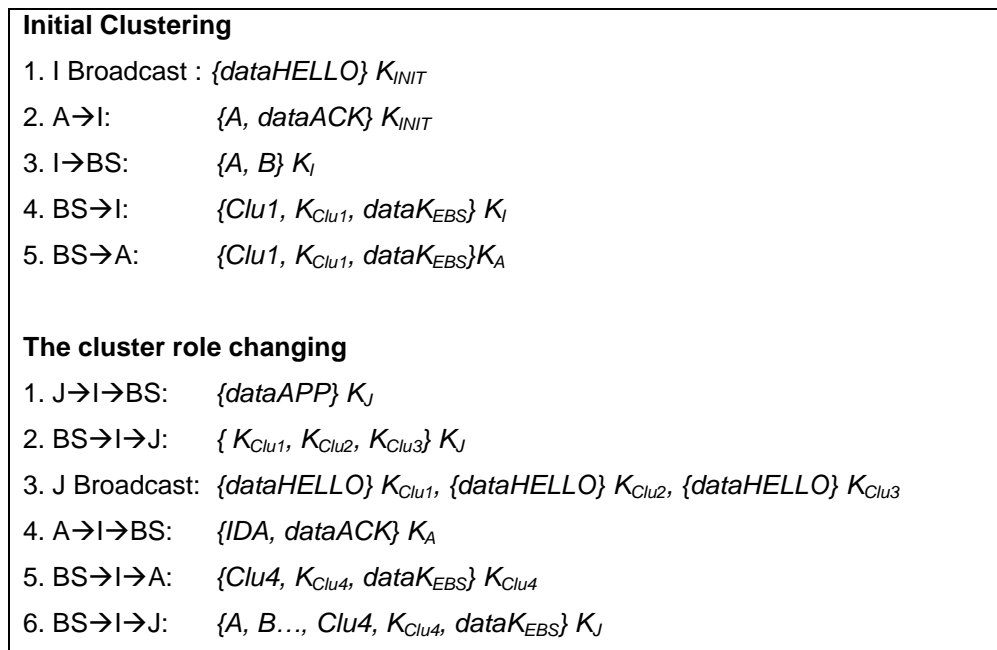
(Figure 3.4 assumes that nodes A and B are in I 's cluster). For each cluster the BS generates an ID, a cluster key K_{Clu1} and an EBS management key set information $dataK_{EBS}$. $dataK_{EBS}$ contains the IDs of the keys for the receptor nodes shared by the other nodes within the cluster. BS then sends a message to each node containing the generated information for the node's cluster. These messages are encrypted with the corresponding key shared between each individual node and the BS. Now the clusters are formed, and the global key K_{INIT} is discarded. The Cluster Role Changing Protocol allows re-electing CHs and CMs of WSN nodes to distribute increased power consumption due to CH duties among all nodes. This protocol is operated periodically, as one WSN node cannot act as the CH for too long, otherwise its battery may be depleted prematurely.

Figure 3.4 also outlines the Cluster Role Changing Protocol for a cluster with current head I , new CH J and a node A that wants to change from I 's cluster to J 's cluster. As in the initial clustering protocol, nodes use the LEACH algorithm to elect new CHs. The new CH J sends an application message via the current CH I to the BS. This message is encrypted with the key shared between J and the BS. The BS replies (via the current CH I), by sending the keys of three geographically neighboring clusters to node J , which then uses the keys to encrypt and broadcast the HELLO message to the WSN nodes in its cluster and the nearby clusters (Figure 3.4 assumes that J has three neighboring clusters, with the cluster key K_{Clu1} , K_{Clu2} , and K_{Clu3} respectively). As nodes that want to become members of J 's cluster do not share a secret key with J , they send an acknowledgement message via their current CH to the BS. These messages are encrypted with a respective key shared between the node and the BS.

When all acknowledgements are received, the BS generates new cluster keys (K_{Clu4}) and new EBS management key sets information $dataK_{EBS}$ for all members in the new cluster $Clu4$. Similar to the initial clustering protocols, the BS then

sends a message to the nodes containing the generated information for the node's cluster. Finally, the BS sends a message to the new CH that contains the IDs of all nodes in the corresponding cluster, encrypted with the key shared between the CH and the BS.

CBKM is not a secure protocol. I have found its security flaw and published a revised protocol in [DZC08].



Added Legend:

I, J:	CHs	A, B:	CMs
CluX:	the ID of Cluster X	BS:	Base Station
K_{INIT} :	Initial key for first cluster establishment	K_Y :	Key of Node Y
K_{CluX} :	Cluster X's cluster key	dataHELLO:	HELLO message
dataACK:	Acknowledgement message	dataK _{EBS} :	EBS key set data
dataAPP:	Application of electing as CH		

Figure 3.4 CBKM Protocol Specification

3.6 SecLEACH

SecLEACH is a security enhancement of LEACH, which aims to fix the vulnerabilities of LEACH [OFV07]. While SecLEACH inherits the basic idea from LEACH, it has additional communication steps related to security enhancement. The security protection used in SecLEACH, such as encryption, key update, and message authentication code (MAC), will be introduced in Chapter 5.

The protocol establishes a large pool of session keys and their node IDs. The protocol pre-distributes the keys and their IDs to nodes by using a pseudo-random scheme. Every node gets a sub-group of keys for communicating with other nodes. The protocol also creates a one way hash function F and a session key K_0 . F can be used to deduce the next session key: K_1, K_2, \dots, K_m . K_0 is pre-stored in every node. The series of keys $K_0, K_1, K_2, \dots, K_m$ in the key chain is used to communicate and loose-synchronize with BS. We assume initially every node obtains the sub-group of session keys and their IDs, and the first key of key chain K_0 . Every node knows the one-way hash function F and which keys it shares with any other nodes. Also, the protocol establishes a pair-wise key for every node which directly communicates with BS.

The protocol must guarantee the secrecy of the key pool: every single key should be only stored by authorized nodes. Every node obtains the information of possession of keys of other nodes only by their keys' IDs. The protocol must guarantee the secrecy of the one-way hash function: only BS and authorized nodes know the function. The protocol must guarantee the authenticity of messages. Once the message is signed by a node with MAC function, only the node it wishes to send can deduce the same MAC digest as it sends, thus ensuring the message is authenticated. The protocol must guarantee the freshness of the messages.

3. Review of Wireless Sensor Network Routing Protocols

In SecLEACH, the deployment of the network is more complex than LEACH. The protocol assumes the BS has the knowledge of all nodes in the network. According to the amount of nodes, BS generates a large pool of keys and their identities prior to deployment. The keys and their identities are generated pseudo-randomly.

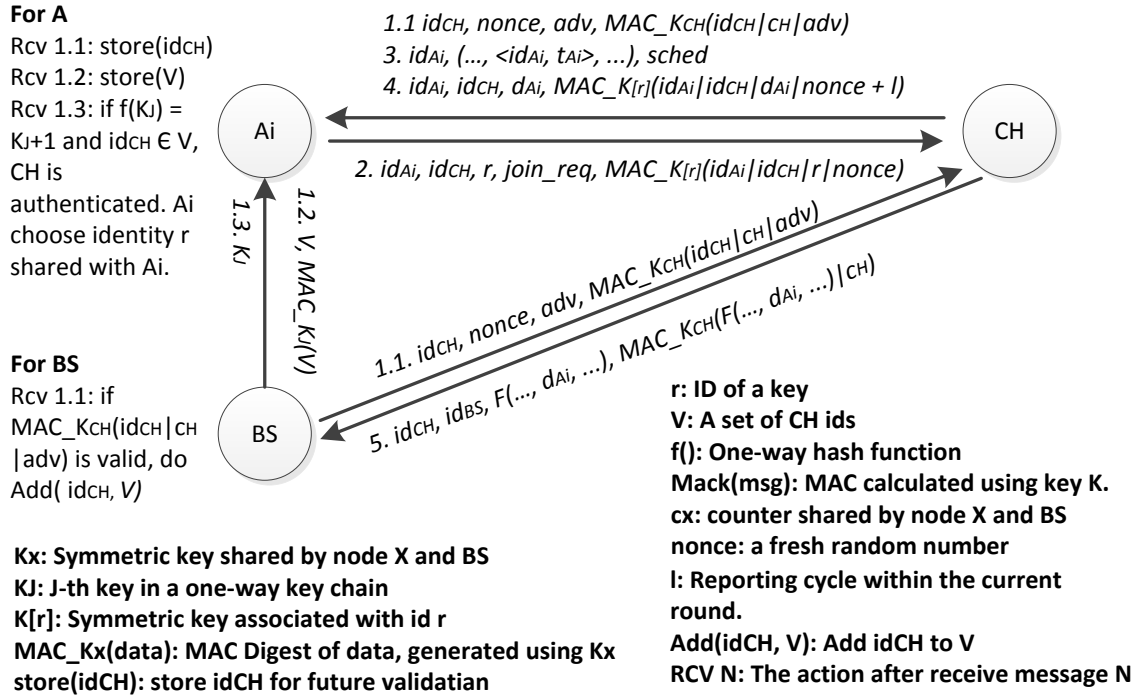


Figure 3.5 SecLEACH Protocol Description

The keys are for further node-to-node communication. This generation process is performed only once during the network's life time. BS should ensure every node can share certain keys with some other nodes for node to node communication. BS then distributes the keys to every node. At the same time, the protocol needs to ensure a certain percentage occurrence of two nodes being able to share the same key for communication. Then, prior to deployment, the BS assigns the pair-wise key to every node and the start key of key chain K_0 .

SecLEACH has the same procedure for cluster setup and CH election as LEACH. In extra, in step 1.1, the CH generates a nonce, also called random number, and a MAC digest using the pair-wise key shared by CH and the BS, to include in the initial message. Only the BS can verify the message to prove its authentication (BS recreates the MAC digest) and freshness (BS shares the counter value with CH). Each ordinary node (in Figure 3.5, it is node A_i) stores the CH's ID. Once BS verifies the MAC message, it then adds the CH's ID to the list of valid IDs. BS broadcasts the list to every node by using the currently valid key K_j in the key chain. After that, BS broadcasts K_j to the network, to inform every node that the current key K_j is not valid anymore (step 1.3). Another reason for broadcasting K_j is for synchronization. If node N missed the last two messages and it possesses K_{j-2} , it cannot verify the message sent in step 1.2. Once it receives K_j , it can calculate $f(f(K_{j-2})) = K_j$. Once it calculates K_j , it knows the previous unverifiable message is valid and sent by BS. At the same time, if the message 5 contains CH's ID, it also knows the message in step 1.1 was sent by the CH.

In step 2, the ordinary node A_i decides which CH it will transfer data to. The decision is based on the signal strength sent in step 1.1, and if they share the same key. Once it decides which CH it will transfer to, it sends the CH its ID, the CH's ID, a join request, and a key ID, which is chosen from the key sets it shares with the CH. Another party will not gain any more knowledge than it already has as only the ID of the message is sent by plain text, not the key itself. The message is signed by the key $K_{[r]}$ and the nonce CH transferred in step 1.1 is included. The CH verifies the message, and then schedules the time slot for each node which will transfer data to it (step 3). The result is broadcast to the network. Therefore, every node knows when to transfer data.

The set up phase is finished and the steady-state phase starts. When the time comes, node A_i transfers 4th step message. The message includes A_i 's ID, the

3. Review of Wireless Sensor Network Routing Protocols

CH's ID, and its data; and is then signed by MAC, which includes the nonce incrementing l , in order to prevent replay attack. After the CH receives data from all nodes, it compresses the data and sends it to BS, and also signs and includes the counter value (step 5).

4. Security of Wireless Sensor Networks

Security protection is essential when confidential data is involved in the WSN applications. WSN has the same security requirements as the traditional network, but faces more threats due to their limitations. This chapter discusses security requirements of WSN routing protocols and outlines general attacks against these protocols.

4.1 Security Requirements of WSN Routing Protocols

Security can be treated as a separate component in the system's architecture, which is a potentially serious risk in network systems, especially in WSN routing protocols [SKS11]. The security services of WSN should be able to protect the information and resources from attacks and misbehaviour [WAR06]. The WSN routing protocols' security requirement is similar to the information security [Vac09]. The security requirements in WSN include [WLS06] availability, integrity, confidentiality, authenticity, and freshness.

4.1.1 Availability

The network service should be available when it is needed. The system that provides the service must be functioning correctly in all its components, which includes the computing systems used to process and store the information, the

communication systems used to access the system, and the security control used to protect the system. The service with high availability requirements should remain available at all times, and be fault-tolerated for power supply outages, hardware failure, and system upgrade. Ensuring availability includes the ability to fight against a denial-of-service attack.

4.1.2 Integrity

The desire network system should be able to detect or prevent the modification to data without authorization [SaK09]. This involves authorized users' accidentally, or adversary's intentionally, modifying the data, either during transmission or while being stored in the node's storage space. Violation of integrity may result in disasters to the WSN. For example, modification of destination node address makes the receptor node miss the packet, or the modification of the control information disorders the receptor node's configuration, and possibly disables the receptor node.

4.1.3 Confidentiality

Confidentiality is defined as "ensuring that information is accessible only to those authorized to have access" [SaK09]. Confidentiality is the term used to prevent the disclosure of information to an unauthorized party. For example, a Medical WSN application with patient's physical condition, information monitoring and report functionality must prevent the patient's physical condition information being leaked to an adversary, or a Military WSN application with friendly force information exchange capability must not expose any information to an adversary,. The common way of ensuring confidentiality is using data encryption technologies. Due to the limited battery capacity and

relatively weak processor capability characteristics of a WSN node, the public key encryption algorithms are too expensive for WSN applications [Karlo03]. Currently, the symmetric key encryption algorithms are the main confidentiality assurance method. Breaches of confidentiality have many forms in WSN: an adversary capturing the messages can compromise a certain number of nodes to gain stored sensitive information, or may pretend to be the base station or legitimate nodes to gain the valuable information.

4.1.4 Authenticity

In information security, the authenticity means both parties in the communication can validate their identities in WSN, allowing the receptor WSN node or BS to ensure that the received message is actually sent by the legitimate node. Although WSN are objective oriented, focusing on the whole network's cooperation instead of single nodes, verification of the node identity is important for security protection. The encryption of known content to both parties or MAC (Message Authentication Code, will be discussed in Chapter 5) can be used to protect the messages' authenticity.

4.1.5 Freshness

Freshness implies that the received message is up-to-date and not replayed by the adversary prior to the protocol run. Although the WSN nodes in WSN often are not strictly time-synchronized, ensuring the messages are recent is as important as recording the environmental condition changes over time.

4.2 Attack against Wireless Sensor Networks

WSN suffers from various attacks. Due to similarities in the communication systems, attacks against traditional wireless ad hoc networks also apply to WSN [CMY10]. Further, attacks such as Hello flood [SJS10], wormhole [HPJ02], sinkhole [Kro08], Spoofing [KaW02], and Sybil [Uou02] target specific limitations of WSNs.

Security flaws contained in WSN routing protocols can impact on other parts of the application. For example, in [DLC08], the authors revealed multiple weaknesses in the Cluster Based Key Management Protocol (CBKM) [SFQ08], where the adversary can utilize these weaknesses to:

- Impersonate the BS and as a consequence read all sensing data or disturb the network structure
- Impersonate CMs to send incorrect sensing data to the BS.

Thus, CBKM fails to protect the messages' authenticity (the malicious node can impersonate either the BS or CMs without being noticed), freshness (the replayed message cannot be detected), confidentiality (sensed data can be read by the adversary), and integrity (the receptor node cannot detect the modification to some messages, such as the 2nd message in the cluster role changing phase).

4.2.1 Attacks Breaching Confidentiality

The broadcasting nature of the wireless communication in WSN makes eavesdropping reasonably easy to achieve. If the messages are not encrypted, they are directly exposed to any adversaries that are in range of the radio

communications. Further, the hostile deployment area and the unattended operating nature means the nodes may be captured by an adversary. The adversary can learn valuable information from the captured messages or compromised WSN nodes, such as network topology, position of the base station and session keys.

4.2.2 Attacks Breaching Integrity

Adversaries can alter the content of messages in order to disorganise the routing arrangement or modify the sensed data. This form of attack can be easily launched if messages are sent in clear and no integrity protection scheme is used.

4.2.3 Attacks Breaching Freshness

The adversaries can retransmit the old recorded message to the network. If the nodes recognize the message as legitimate, many problems may arise: an adversary may trick nodes into using old compromised session keys; the iterated assignment of cluster head role to the same node may cause it to deplete its battery quickly, etc.

4.2.4 Attacks Breaching Authenticity

In WSN, attacks which breach authentication have two general forms: Firstly, a malicious node may pretend to have a specific role in the network, e.g. a cluster head or BS, in order to deceive other nodes into sending it data. Secondly, an adversary may pretend to be the nearest neighbour of multiple nodes. This may result in the adversary being chosen as the router node by these nodes.

4.2.5 Typical Attacks

Replay Attack

A typical attack breaching freshness is the replay attack. In replay attacks, the adversary eavesdrops on the message and stores it, and then transmits it at another time. By launching such an attack, the adversary can pretend to be the message originator node and gain the other nodes' trust.

Wormhole Attack

Different from the replay attack, wormhole attack [HPJ02] tunnels a message, transmits with a low latency media (e.g. Ethernet cable), makes the packet arrive at the other side of the network faster than it should, and possibly with stronger signal strength. This could convince a node with multiple hop distance to BS to believe it requires only a single hop transmission to BS, or to believe the distances between it and the sender is much closer than it actually is.

Figure 4.1 is an example of a wormhole attack. In Figure 4.1, two malicious nodes tunnel into the packet in order to convince node A that the communication with BS requires only a single hop.

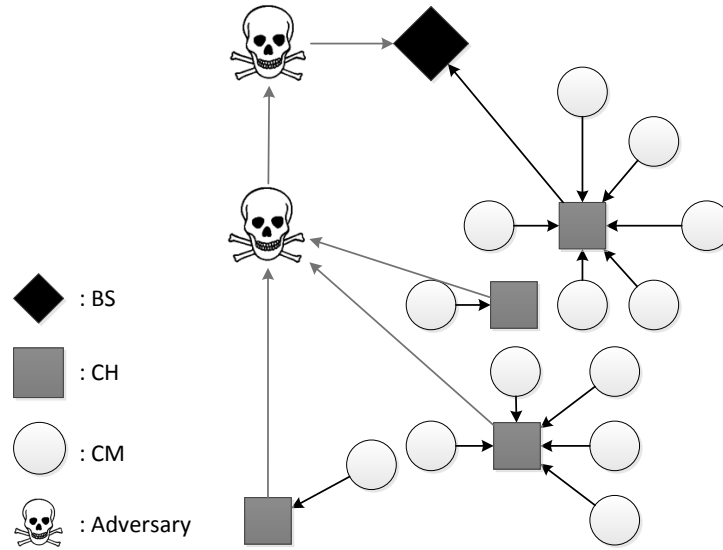


Figure 4.1 Sample of Wormhole Attacks

Sybil Attack

Sybil attack [Dou02] (shown in Figure 4.2) is a representative attack breaching authentication. In Sybil attack, a malicious node pretends to be multiple WSN nodes by presenting their IDs. Such an attack can remarkably disturb the network's efficient routing and data storage [KaW02].

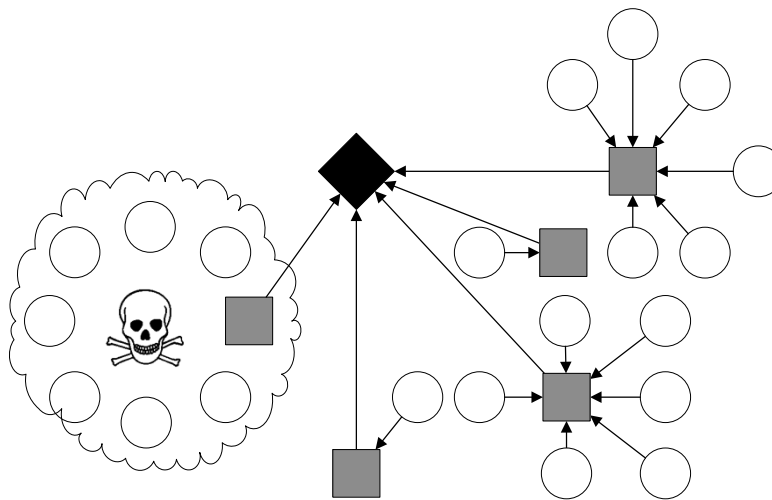


Figure 4.2 Sample of Sybil Attack

HELLO Flood Attack

HELLO flood attack [HRH06] is another example of an authentication attack (shown in Figure 4.3). WSN nodes often identify their neighbors self-consciously by broadcasting a Hello message. The neighborhood recognition process is normally based on the assumption that every node transmits the Hello message with the same signal strength. By utilizing this characteristic, the adversary uses a laptop class node to broadcast the Hello message with sufficient power, to convince the nodes to be their nearest neighbor.

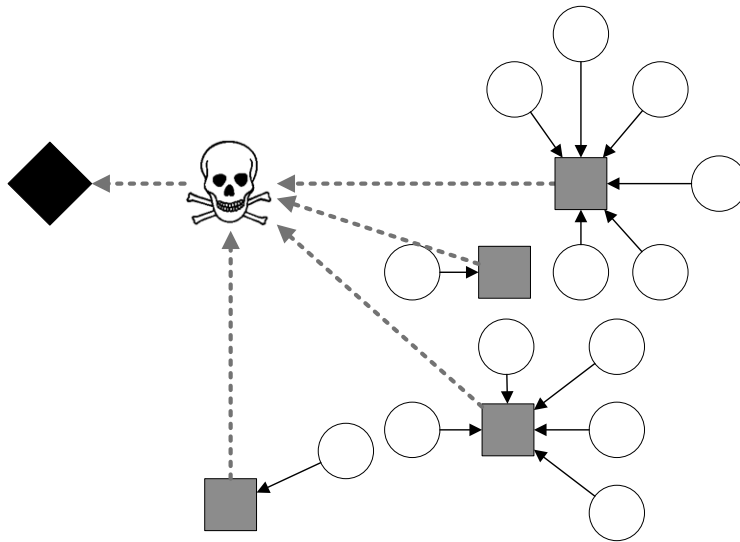


Figure 4.3 Sample of HELLO Flood Attack

Other Attacks

Selective forwarding attack: it is an attack aimed at the WSN's network topology. For launching the attack, two preconditions have to be satisfied: first, the nodes believe their neighbors forward all the packets they transmit. This is the basic assumption of WSN; second, a malicious node is placed in a routing path of a multi-hop WSN and as a result gains trust from its neighbors. Then the adversary can first launch a HELLO flood attack to achieve this precondition.

After this, the malicious node can be used to refuse to forward some or all packets through it [Bys11].

Sinkhole attack: the attack utilizes the characteristic of WSN in that all nodes forward packets to the same destination (we assume the network has only one BS), placing a malicious node in the routing path to attract as many nodes as possible in order to transmit packets to it [NLL06]. The malicious node convinces the other nodes it is efficient in forwarding packets to BS through it (which is not difficult when the malicious node is a powerful laptop class node). Then the adversary can simply drop all the packets or launch other attacks.

Node compromise: Once a node is captured, the adversary can compromise and re-deploy it into the network or replace it with a more powerful device (i.e. a laptop). The compromised node can be the original node with malicious code running, or a laptop class node with the compromised node's ID (which is more dangerous). The compromised node helps the adversary launch other attacks such as Sybil and Sinkhole attacks.

4.3 Security Analysis of the BCDCP Protocol

An unprotected "plain" WSN routing protocol will have to face a large amount of attacks when it is targeted by an adversary. BCDCP has been evaluated to use 11% and 15% less energy than PEGASIS and LEACH, respectively [MMB05]. Thus, I am going to discuss the potential attacks that can be mounted against the BCDCP protocol, as an example to show what kinds of attacks the unprotected WSN routing protocols may suffer from.

4.3.1 Attacks Breaching Confidentiality of BCDCP

In BCDCP all messages are sent in plaintext and an adversary can easily obtain the sensed data and the routing information of the current round by simply installing a receptor antenna in the network. Thus, the BCDCP protocol is not applicable to the applications that require sensed data to be kept private.

All messages in BCDCP contain valuable information, such as the node's location and remaining energy information in the first step of the setup phase, the network topology information in the second and third steps. Also, some applications may require that the sensing data information in the data communication phase is kept confidential, e.g. in traffic control, home automation and healthcare applications.

4.3.2 Attacks Breaching Integrity of BCDCP

In BCDCP, neither the WSN nodes nor the BS can detect illegal modification to the data. The adversary can simply modify the routing information or the sensed data to infest, or even worse, to disable the application. For example, in step 1 of BCDCP's setup phase, the adversary can modify either the location ID or remaining energy of the message to mislead BS in its clustering and routing decisions. The adversary can also modify the CH-CH routing information as well the member IDs of a cluster in step 2 of the setup phase. The former can be used to route all data packets through a compromised node or a malicious node (as shown in Figure 4.4 (a)). The latter can be used to isolate nodes from the network (as shown in Figure 4.4 (b)).

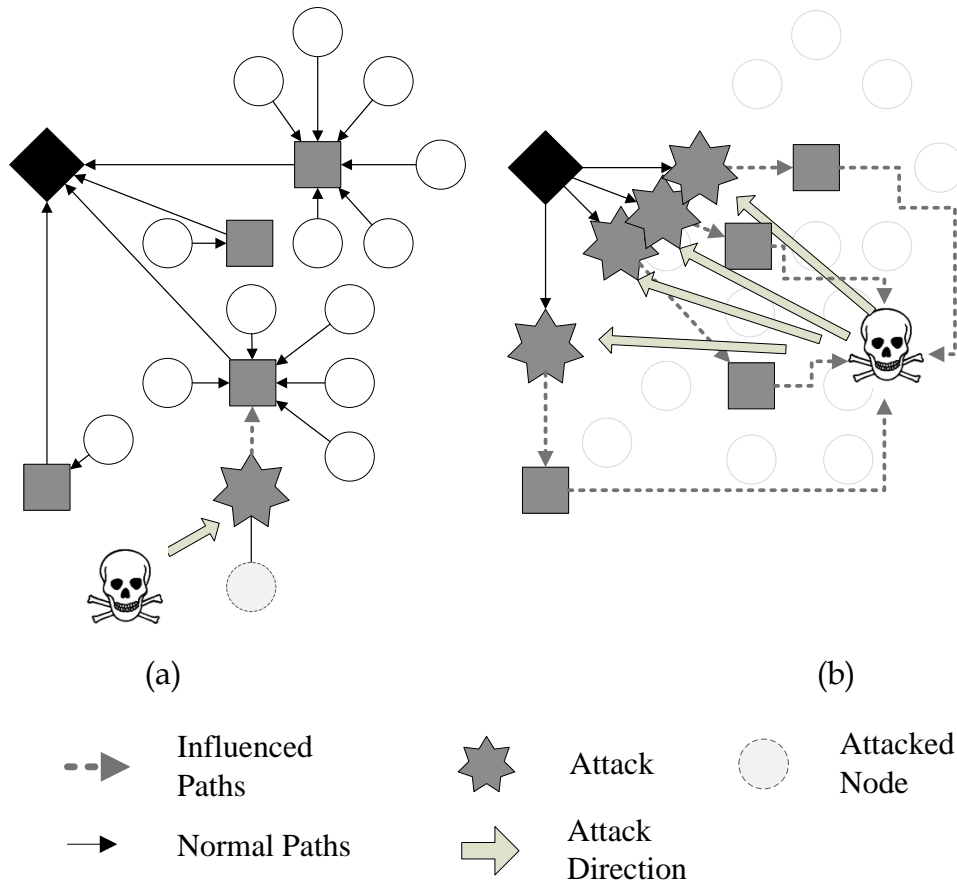


Figure 4.4 Examples of Attacks Breaching Integrity in BCDCP

4.3.3 Attacks Breaching Freshness of BCDCP

In BCDCP, since the protocol has no protection for detecting replayed message, the adversary can transmit the messages recorded in the previous rounds, and the WSN node / BS would still recognize them as fresh. E.g., in the first step of the setup phase, the adversary can simply replay the message sent by a node. For example, if node A's message in step 1 is repeatedly replayed then the BS would assume that A's remaining energy remains the same. Thus, the BS will continue to allocate A as a CH. Consequently, A's battery would be drained fast (as shown in Figure 4.5).

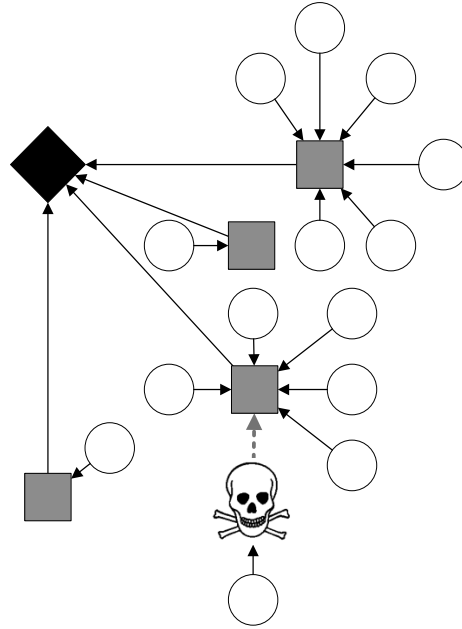


Figure 4.5 Examples of Attacks Breaching Freshness in BCDCP

4.3.4 Attacks Breaching Authenticity of BCDCP

In BCDCP, the WSN nodes or the BS have no knowledge of whether the incoming message comes from a legitimate node. Therefore, the adversary can easily pretend to be a legitimate node(s) and / or the BS to transmit any message in BCDCP's 5 steps. The receptor node would as a result have no idea if the incoming message is sent by the legitimate node. For example, the adversary can act as BS in assigning CHs, arrange CH-CH routing paths in the setup phase, or pretend to be regular WSN nodes in order to send the BS false sensed data in the data communication phase. Furthermore, the adversary can easily launch a Sybil Attack [Uou02], in order to act as multiple WSN nodes to communicate with the BS.

5. Protecting Wireless Sensor Network Routing Protocols

The attacks listed in chapter 4 reveal the requirements to properly protect the WSN routing protocols. In this chapter, common methods used in the protection of protocols will be introduced. The methods have been adopted by some WSN secure routing protocol designers, and can be useful reference to design new secure WSN routing protocols.

5.1 Countermeasures to the Attacks against WSN routing protocols

To protect the WSN routing protocols, the foremost step should be having knowledge of the countermeasures used to defend against the WSN aimed attacks. In this section, the countermeasures to protect the WSN routing protocols' confidentiality, integrity, freshness, and authenticity are introduced [KaW02].

5.1.1 Countermeasure to Attacks Breaching Confidentiality

Traditional networks achieve confidentiality by utilizing symmetric keys or public keys to encrypt the message. Only authorized parties with correct keys can reveal the original message content. The public key solution is proven to be more secure and more convenient in authentication as well as key exchange, but is less computationally efficient than the symmetric key solution: a MICAZ WSN node requires 0.89ms and 1.13ms to perform single block 128-bit key encryption and decryption operations, respectively [ZDC11], while 18.76ms (21 times than AES encryption) and 48.04ms (42 times than AES decryption) are required to generate and verify a RSA-1024 signature, respectively [PLP06]. For WSN applications, as the microcontrollers have relatively weak computational capability and limited power supply, public key solutions are often too expensive for them [PSW04]. Hence, defenses against attacks breaching confidentiality are usually based on symmetric key solutions [WAR06].

However, once a secured WSN application adopts the symmetric key solution, the key stored in the WSN nodes' memory lack physical protection. The general idea in WSN is not keeping the 'big' secret, such as the global key in the node's storage device. In [DXC06], the authors introduce two keys per node countermeasure in their Secure Cell Relay Routing Protocol. Every node possesses two keys, one for directly communicating with BS, named the private key, and the other for communicating temporarily with the neighbors, called the session key. When BS notices that a node is captured, it simply updates the session key and broadcasts to each WSN node except for the captured ones. This countermeasure may require the sensor hardware's microcontroller to have the capability of encrypting / decrypting the messages, and shortens the battery life due to the extended computation and message overhead.

5.1.2 Countermeasure to Attacks Breaching Integrity

Traditional networks utilize technologies such as digital signature and secret key encryption to ensure message integrity [Moh04]. In WSN, a digital signature is too expensive to achieve, not only because of the more advanced processor capability requirement, but also due to the lack of certificate authority (CA) [NLD08]. Symmetric key encryption is one of the general countermeasures to defend against such attacks, plus it ensures confidentiality in WSN. However, some applications do not require confidentiality protection (e.g. [SWC07]). An efficient countermeasure is to use Message Authentication Code (MAC), presented in [Inf02]. This involves using a keyed hash function and a given plaintext to generate a MAC digest which appends to the message to be transmitted. Only the party who possesses the right key and the original message can regenerate the same hashed content, hence proving the message has not been altered during the transmission. MAC has the capability of proving authentication as well (which will be discussed in chapter 5.3.4) and has been adopted by many security routing protocols [DXC06] [OFV07] [SWC02].

5.1.3 Countermeasure to Attacks Breaching Freshness

Countermeasures to the freshness attack in WSN mainly comprises of adding a timestamp or counter value to the message. A timestamp often requires the network to be strictly time synchronized. Currently, many time synchronization protocols have been proposed ([GKS03], [MKS04], [Ste97]). However, time synchronization of all WSN nodes often implies extra energy consumption for transforming and receiving sync messages.

Most WSN routing protocols are not critical time driven. In [SWC02], the authors presented a loose time synchronization protocol. BS generates an initial key, pseudo-randomly, then uses a one way hash function to generate a series of keys; we can denote them as $K_1, K_2, K_3, \dots, K_n$. Every node is pre-loaded K_n with the hash function prior to deployment. At the end of the time slot BS broadcasts K_t in the key series. For example, at the end of 3rd time slot, BS broadcasts K_3 to the network. Nodes that receive the key will first set value $K_x = K_3$, and then iteratively calculate $K_x = f(K_x)$ and record the number of times of calculation (we denote it p), until they get $K_x = K_n$. Nodes also record the previous calculation times in the last time slot, which we denote as q . If $q - p = 1$, the node's time is correct; if $q - p > 1$, then the node must have missed the last one or more of the synchronization chances.

Counter value is another efficient solution to defend against freshness attacks [OFV07]. The idea is that BS and the node share the same counter value and keep it secret. In every message transmission, both sides increment the counter value by 1. For a WSN routing protocol using encryption and decryption, the counter value is encrypted. For a MAC solution protocol, the counter value is contained in the hashed message, but not in the plain content. A replayed message contains only the out-of-date counter value, which is easily detected. However, the loss of certain messages can lead to the desynchronization of counter values between the sender and receptor nodes, and an attack called *Supress-and-Desynchronise attack*, which aims to take advantage of this vulnerability [LDC11a] [LDC11b]. A certain strategy has to be adopted to help regain the synchronization of the counter value between the two nodes.

5.1.4 Countermeasure to the Attack Breaching Authentication

In WSN, there are two countermeasures to authentication attacks. They are key encryption and MAC. By using encryption, the sender node will place identifiable bytes (e.g. the sender node's ID) into a predefined location of the message which is to be encrypted. The receptor node that decrypted the message will check the bytes in the location. If the bytes are recognizable, the message is authenticated.

By using MAC, the integrity of the message can be verified by regenerating the hashed content, which can be made only when the node or BS possesses the right key. This also verifies the one who generated the message prior to the transmission being authorized [Ste97]. In contrast, the adversary, who does not possess the right key and precisely correct message content, cannot generate the correct message hash digests.

5.2 Block Ciphers

In this section, a number of block cipher algorithms that are suitable for securing WSN routing protocols will be introduced.

5.2.1 RC5

RC5 is a block cipher designed by Ronald Rivest in 1994 [Riv94]. The term "RC" stands for Rivest Cipher. The unique characteristic of RC5 differential in regards to other block cipher algorithms is its variable block size (32, 64, or 128 bits), key size (0 - 2040 bits), and number of rounds (0 - 255 rounds). Each round is structured as a Feistel-like network, shown as Figure 5.1.

RC5 utilizes heavily data-dependent rotations: the amount of rotation is dependent on the input data, and is not predetermined. RC5 algorithm is comprised by a number of XOR operations and modular additions. As Figure 5.1 shows, the algorithm of one round is quite simple. The security of the cipher-text intensifies in regards to the number of rotation rounds.

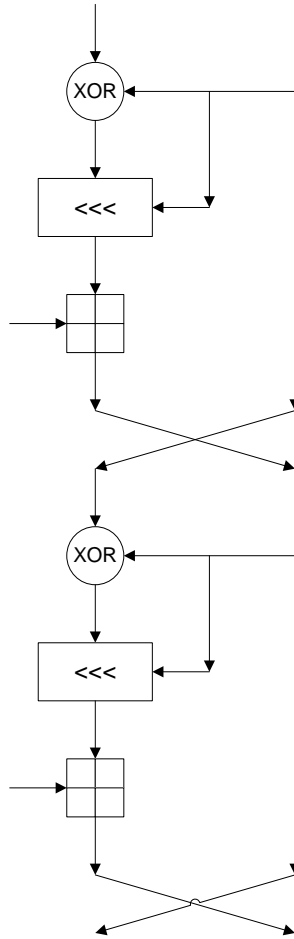


Figure 5.1 RC5 Algorithm Structure

5.2.2 Advanced Encryption Standard (AES)

The U.S. government adopted the Rijndael algorithm [DaR99] as the new Federal Advanced Encryption Standard (AES), which was published by the National Institute of Standards and Technology (NIST) under U.S. FIPS PUB 197 (FIPS 197) in 2001 [Fed01].

AES's design principle is based on a Substitution permutation network. AES supports a fixed block size of 128 bits and variable key sizes of 128 bits, 192 bits, and 256 bits. The AES cipher initially organizes the plaintext into a 4x4 array of bytes (called the state) and expands the key into the required number of round keys. Then it performs the initial round followed by a number of repetitions of transformation rounds and a final round (cf. Figure 5.2). The number of repetitions depends on the key-size: for a 128-bit key 10 rounds are used, for a 192-bit key 12 rounds are used and for a 256-bit key 14 rounds are used. A set of reverse rounds using the same (expanded) encryption key is applied to transform cipher-text back into the original plaintext.

Except for the initial and final round, each round contains the following steps:

- **AddRoundKey**: combining the round key with the state.
- **SubBytes**: Each byte is replaced with another according to Rijndael's S-Box.
- **ShiftRows**: Shifting each row of the state with a certain offset. The first row remains the same in Rijndael.
- **MixColumns**: Combining the four bytes in each column using an invertible linear transformation.

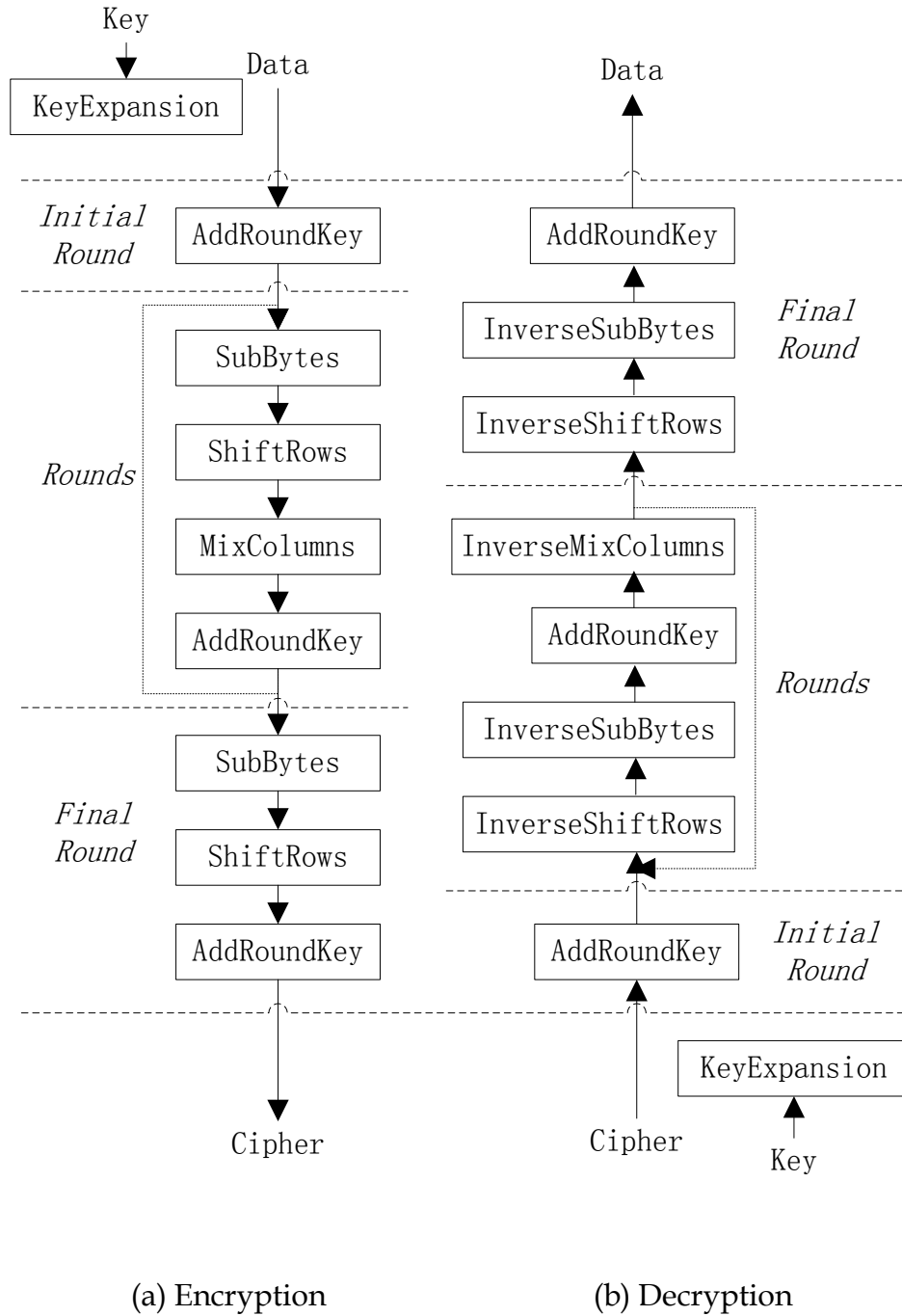


Figure 5.2 AES Encryption and Decryption Algorithm Structure

The initial round performs only the step *AddRoundKey*, while the final round does not perform the *MixColumn* step.

For many years, RC5 [Riv94] were considered to be one of the most suitable cryptographic ciphers for WSN applications [KaW02] [ViP07]. Recently AES was proposed as a preferable option [LDH06] [HNL07], as it provides stronger security protection while still being suitable for the low resource requirements on WSN nodes. AES encryption with a 128-bit key has been verified to be secure enough to protect classified information up to US Government's SECRET level [Hat03]. While some theoretical weaknesses have been found in AES - e.g. the XSL attack against 128-bit AES [CoP02] - these require the order of 2100 operations and, thus, are regarded as not practical with current technology. 128-bit key AES is considered as sufficient for WSN applications [HNL07].

The performance of the AES algorithm has been widely analyzed [XCS06] [ZhN08]. For example, the original Rijndael algorithm contains some redundancy in the shifting of rows and mixing of column operations. As these operations are independent of the state, they can be replaced by a sequence of table lookups and combined with the SubBytes operation to obtain a more efficient algorithm [BBF03] [ZhN08].

5.2.3 Block Cipher Modes of Operations

All block cipher algorithms allow the encrypting of only a single block of data (most algorithms' block sizes are fixed - except for RC5) at a time. The message which is shorter than the block size needs to be extended, and the one which is longer than the block size has to be separated into multiple blocks of data, with potential padding of the last block (unless the message size is equal to multiple

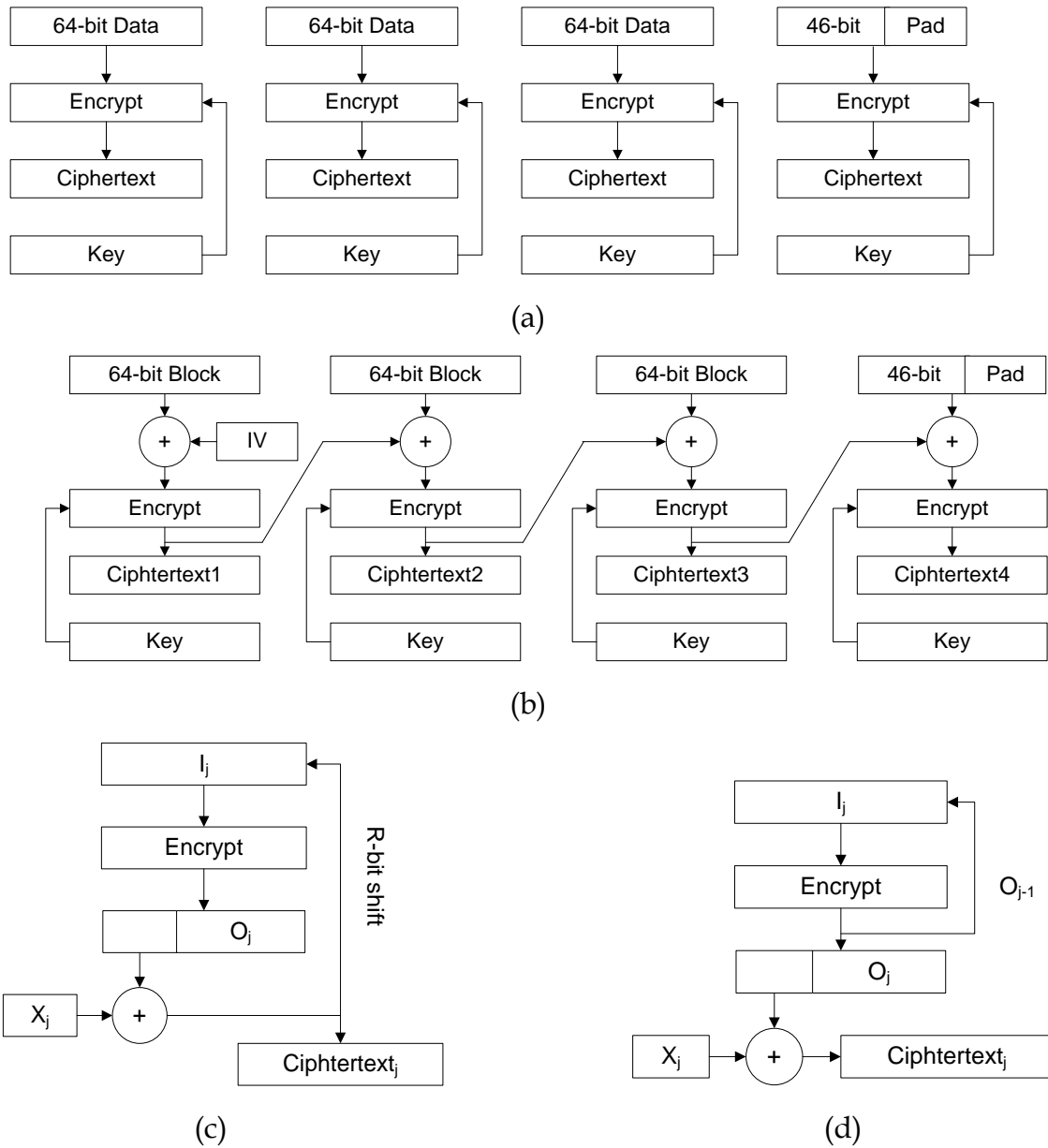
times the block size). There are a number of operations which are utilized to deal with the long message encryption: Electronic codebook (ECB), Cipher-block chaining (CBC), Cipher feedback (CFB), and Counter (CTR) modes.

ECB is the simplest mode of operation. The message is separated into blocks and each block is encrypted individually (Cf. Figure 5.3 (a)). Encryption of each block is independent of each other. ECB has a disadvantage in that the encryption result of identical plaintext block is identical. For example, while obtaining the encrypted BMP image, the adversary can distinguish the image from the cipher-text stream directly. In order to adapt to the situation, the following modes were announced [Dwo01].

CBC Mode (Cf. Figure 5.3 (b)): Each block of ECB encrypted cipher-text is XORed with the next plaintext block to be encrypted, thus making all the blocks dependent on all the previous blocks. The First block is XORed with a 64-bit number called the Initialization Vector.

CFB Mode (Cf. Figure 5.3 (c)): A block of plaintext less than 64 bits is applicable to this mode. A 64-bit cipher-text is obtained by encrypting an arbitrary value (stored in shift register). The left most M bits are selected to XOR with the real plaintext to output the cipher-text. The cipher-text is fed into the shift register, used as the plaintext for the next round.

OFB Mode (Cf. Figure 5.3 (d)): Similar to CFB, the only difference is that the encryption output is fed to the shift register instead of CFB's XORed cipher-text.



a. ECB Mode, b. CBC Mode, c. CFB Mode, and D. CTR Mode

Figure 5.3 Modes of Operations

5.3 Message Authentication Code

Message Authentication Code (MAC) is a piece of information used to authenticate a message. MAC is designed to protect the message's authenticity as well as its integrity by allowing verifiers (who also possess the same secret key)

to detect any changes to the message content. To authenticate the message, the message sender generates a piece of information by hashing or encrypting the message-to-sent using a secret key. The hashing results is called MAC digest [BCK96]. The unauthorized parties should be unable to find the message from the digest, modify the message without changing the digest, or find any two messages from the same hash or encryption [Dwo05]. In this chapter, we introduce two popular MAC algorithms: Hash-based Message Authentication Code (HMAC) and Cipher-based Message Authentication Code (CMAC).

5.3.1 Hash-based Message Authentication Code

Hash-based Message Authentication Code (HMAC) provides a way to check the integrity and the authentication of information transmitted over an unreliable medium by using an approved cryptographic hash function in combination with a secret key to calculate a message authentication code (MAC) [BCK96]. HMAC is designed to be able to use the security keys and the currently available hash functions, and easily replace the underlying hash function for more secure ones later.

To compute a MAC digest, the HMAC operation is performed as follows:

$$\begin{aligned} & \mathbf{MAC}(text)_t \\ &= \mathbf{HMAC}(K, text)_t \\ &= \mathbf{H}((K_0 \oplus opad) \parallel \mathbf{H}((K_0 \oplus ipad) \parallel text))_t \end{aligned}$$

Where K is the secret key shared between the originator and the intended receiver(s), K_0 is the key K after the processing to form a key with an equal length to the hash function's input block size. H is the approved hash function, and

opad and **ipad** are the outer pad and inner pad of the algorithm, both of which have pre-defined values.

Firstly, the key K shall be expanded or shrink to the size equal to the hash function's input size. The processed key is called K_0 . Then, K_0 is XORed with **ipad**, and appended to the data stream text. The hash function is then performed on the appended stream. A similar operation is performed on K_0 and **opad**. The leftmost t bytes (t is the number of bytes of MAC) of the result are chosen as the final HMAC output. Moreover, the HMAC output can be truncated for complying with a low bandwidth channel (such as WSN).

MD5 (Message-Digest algorithm 5) is one of the most widely used cryptographic hash functions with 128-bit hash values [Riv92]. An MD5 hash is normally expressed as a 32-bit number. In order to perform a MD5 algorithm, the message to be hashed is firstly expanded to a fixed length, and then is followed with a 4-round hash operation. However, the collision search attack against MD5 [Ste07] do not compromise the use of MD5 within HMAC [Dob96]. MD5 is currently one of the most widely used cryptographic hash functions in HMAC.

SHA-1 is another widely used cryptographic hash function [EaJ01]. It is different from MD5, SHA-1 in that it has a block size of 512 bits and produces a 160 bits digest. The input message size ranges from 1 to $2^{64} - 1$ bits.

5.3.2 Cipher-based Message Authentication Code

Unlike HMAC, Cipher Based MAC uses a secret key and block cipher instead of a cryptographic hash function as the cryptographic primitive. In the same way as

HMAC, a Cipher Based MAC's change to any plaintext bit will cause different MAC digests.

Cipher Block Chaining MAC (CBC-MAC) [Fed85] is one of such MAC methods. The idea of CBC-MAC is to generate a chain of block ciphers, and each block is dependent on the previous cipher block, as shown as Figure 5.4. In the figure, the message m is divided into blocks named $m_1, m_2, m_3, \dots, m_x$. m_1 is firstly XORed with the zero initialization vector, and then is encrypted with key k . The result is then XORed with the next message block. The procedure iterates until the last block m_n is XORed and encrypted. The final result is the MAC output.

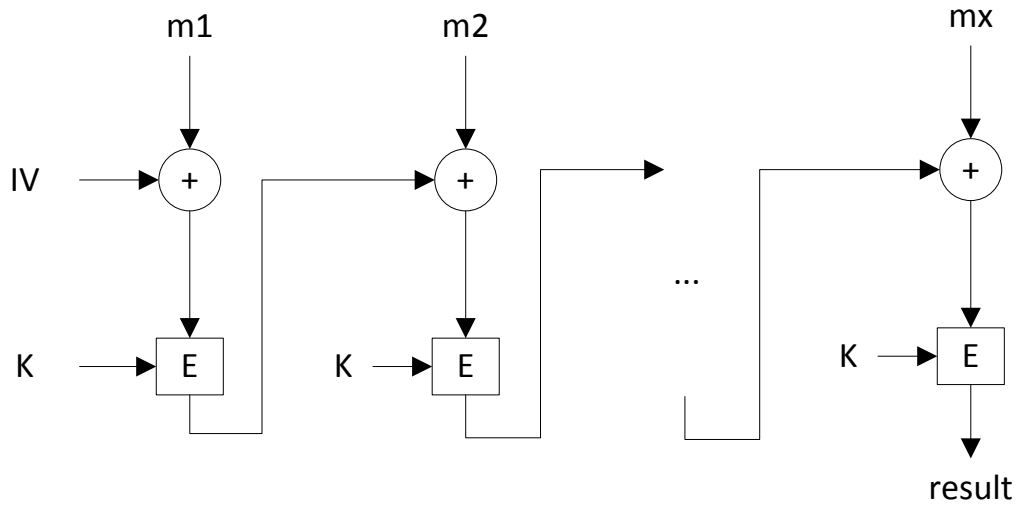


Figure 5.4 CBC-MAC Flow Chart

Cipher-based MAC (CMAC) is the variation and fixed security deficiency of CBC-MAC [Dwo05]. Different to CBC-MAC, CMAC needs to generate two sub-keys k_1 and k_2 using the encryption key k . By dividing the message m into blocks $m_1 | m_2 | m_3 \dots | m_{x-1} | m_x'$, where $m_1, m_2, m_3, \dots, m_{x-1}$ are complete blocks. When m_x' is a complete block, for each block do $m_x = k_1 \oplus m_x'$, otherwise m_x' is filled to a complete block, and then is XORed with k_2 .

Either CBC-MAC or CMAC requires a block cipher algorithm to be the cryptographic primitive. Advanced Encryption Standard (AES) is one of the most suitable choices. Its design principle is based on a Substitution permutation network [DaR99]. AES supports a fixed block size of 128 bits and variable key sizes of 128 bits, 192 bits, and 256 bits. For performing an AES computation, the plaintext is firstly organized into a 4x4 array of bytes (called the state). The key is also expanded into the required number of round keys. Then it performs the initial round, followed by a number of repetitions of transformation rounds and a final round. A set of reverse rounds using the same (expanded) encryption key is applied to transform cipher-text back into the original plaintext.

5.4 Security Protection's Impact to WSN Node Energy Consumption

Upon adding security to a WSN routing protocol, the following extra energy may have to be consumed:

- In the absence of hardware security support, the processor unit may have to work longer to realize the extra security, such as encryption / decryption [HNL07] or MAC generation [KaS06].
- The radio transceiver may have to consume extra energy for transmitting / receiving extra bits of the extended message overhead, caused by adding the security [OFV07].
- There may be hardware security provided either by the processor unit, i.e. Atmel's ATXMEGA series microcontrollers [Atm03], or by the radio transceiver, like Chipcon CC2420 radio transceiver [Chi04]. Either way, they have to be powered for longer in order to finish the computation task.

5.5 Theoretical Estimation of Energy Consumption of WSN Routing Protocols

As we stated in Chapter 4, securing WSN routing protocols is necessary when sensitive information is involved, e.g. in [RoM04] [PHC06] [PSW04] [SKS11] [ShP04]. Once security protection is required, people may want to know the energy consumption performance of the WSN routing protocols before they are adopted. Currently, a number of theoretical energy consumption estimation models are available. In this section, two popular such models will be reviewed.

5.5.1 Xiao's Model to Estimate Energy Consumption of WSN Microcontroller

To evaluate the energy consumption of a WSN routing protocol, the power consumed by the microcontroller, while running routing computations, was always ignored [MMB05] [OFV07]. This was comprehensible for unprotected protocols as the routing computations are relatively simple and do not require much time. However, the security computations involved in the secured WSN routing protocols require large amounts of time and energy [ZDC11] [HNL07] [DAB08]. Neglecting its energy consumption may lead to inadequate or incorrect evaluation results.

Xiao et al. found a way to simulate the actual processor overhead while encrypting and decrypting a block of data using AES algorithm [XCS06]. In their research, they analyzed the algorithm of AES, and deduced the minimum number of processing cycles required for performing an AES encryption and decryption, shown as Eq. 5.1 and Eq. 5.2.

$$\begin{aligned}
 T_E = & (8BT_{and} + 4BT_{or}) + \\
 & [46BT_{and} + (31B + 12)T_{or} + (64B + 96)T_{shift}](R - 1) + \\
 & (8BT_{and} + 7BT_{or} + 3BT_{shift})
 \end{aligned} \tag{Eq. 5.1}$$

$$\begin{aligned}
 T_D = & (8BT_{and} + 4BT_{or}) + (8BT_{and} + 7BT_{or} + 3BT_{shift}) + [161BT_{and} \\
 & + (122B + 12)T_{or} + (32B + 96)T_{shift}](R - 1)
 \end{aligned} \tag{Eq. 5.2}$$

Where T_E and T_D denote the minimum number of processing cycles for encrypting and decrypting block data, respectively; $4B$ is the block size (in bytes), R denotes the number of rounds, and T_{and} , T_{or} , and T_{shift} are the number of processing cycles for a processor to perform single byte-wise AND, byte-wise OR, and byte-wise SHIFT operations respectively.

Further, they assumed $T_{and} = T_{or} = T_{shift} = 1$ hold (each hold represents 1 CPU processing cycle), $B = 4$ holds as the AES adopts the block size as 128 bits, and $R = 10$ for 128-bit key size. In conclusion, they calculated that the overheads for encrypting and decrypting a block for 128-bit key are 59us, and 123us for a 100MIPS processor.

5.5.2 Heinzelman's Energy Consumption Estimation on Radio Transceiver

Heinzelman et al. created a mathematical model to evaluate the energy consumption a WSN node's radio transceiver [HCB02]. The model has been adopted by many researchers to evaluate and compare the performance of their routing protocols.

The model is comprised of a few equations to describe the energy cost of transmitting and receiving k-bit data between two WSN nodes with distance r . They are shown as Eq. 5.3 and Eq. 5.4.

$$E_T(k, r) = E_{Tx}k + E_{amp}(r)k \quad (\text{Eq. 5.3})$$

$$E_R(k) = E_{Rx}k \quad (\text{Eq. 5.4})$$

Whereas $E_T(k, r)$ is the total energy consumption of transmitting k-bit data to distance r , and $E_R(k)$ equals the total energy consumption of receiving k-bit data; E_{Tx} and E_{Rx} denote the per bit energy dissipation on transmitting and receiving messages respectively. $E_{amp}(r)$ represents the per bit energy consumption when amplifying the signal, which is acceptable to the receptor node of distance r . $E_{amp}(r)$ can be represented as follows:

$$E_{amp}(r) = \begin{cases} \varepsilon_{FS}r^4, & r \leq r_o \\ \varepsilon_{TR}r^4, & r > r_o \end{cases} \quad (\text{Eq. 5.5})$$

$$r_o = \sqrt{\varepsilon_{FS} / \varepsilon_{TR}} \quad (\text{Eq. 5.6})$$

Where ε_{FS} and ε_{TR} denote the transmission and reception amplifier parameters and r_o is the threshold distance.

Part 2.

**Development of Secure WSN Routing
Protocols and Establishing Energy
Consumption**

6. Securing BCDCP Routing Protocol

The security analysis of BCDCP protocol (presented in section 4.3) reveals the presence of several security weaknesses in BCDCP, which can be exploited in attacks that drain the battery of individual WSN nodes, isolate clusters from the network or modify transmitted sensing data. BCDCP has been evaluated as more energy efficient than LEACH and PEGASIS [MMB05]. Assuming the same security protection is made to LEACH, PEGASIS, and BCDCP, the secured BCDCP should be more energy efficient than the other two secured protocols. So I decided to base the design of new WSN security routing protocol on BCDCP.

In this chapter, I will introduce the design of secure and efficient cluster-based routing protocols. Two new routing protocols, SeBCDCP-MAC and SeBCDCP-ENC, will be proposed. These protocols aim to provide secure routing, while maintaining the energy efficiency and routing properties of the BCDCP protocol. SeBCDCP-MAC, utilizes MAC to provide authentication and integrity protection for messages. The second proposed routing protocol, SeBCDCP-ENC, uses block cipher encryption to additionally provide confidentiality of messages at the cost of higher energy dissipation.

6.1 Securing WSN Protocols VS Energy Consumption

Many routing solutions focus on efficiency alone and ignore security needs imposed by applications. Securing a protocol results in extra energy consumption, as additional energy is required by the processor for extra security computations (e.g. encryption or hashing operations) and by the radio transceiver in transmitting and receiving extra data bytes caused by longer or additional messages. Further, higher hardware requirements (hardware support for cryptography, faster microprocessor, more memory, etc.) may be needed to support the secured protocol. Thus, securing a WSN protocol has a negative effect on the WSN nodes' battery lifetime. The demanding of longer network lifetime, and the requirement for the security protection are often contrary to each other.

However, securing WSN protocols is necessary when they are employed in applications that transmit sensitive information [PHC06] [PSW04] [SKS11] [ShP04]. For instance, the WSN nodes in a battlefield monitoring application may transfer allied armies' locations. The capture of this message may expose this information to the enemy. A routing protocol for applications involving sensitive data needs to ensure the confidentiality, integrity, freshness, and authenticity of the transmitted data [PHC06].

On the other hand, security protection can be added with energy conservation in mind, by evaluating what level of security are required and only adopting minimal features to satisfy these security requirements.

In contrast to traditional wireless ad hoc networks, WSN are application oriented [RoM04]. Different applications have different security requirements: some may

require information to be confidential and protected against modification (e.g. health care applications), others may only need integrity protection (e.g. traffic control) or may require no protection at all (e.g. wildlife monitoring). The different security requirements lead to different security protection schemes that imply different levels of additional energy consumption. For example, using block ciphers to encrypt messages not only leads to increased energy consumption due to increased processor activities, but may also increase the size of messages if the original message size is not a multiple of the block size [Sta07]. On the other hand, MAC has a steady message overhead [BCK96].

6.2 SeBCDCP-MAC Introduction

In SeBCDCP-MAC, every WSN node possesses an individual symmetric key K_x . This key is installed in the WSN node prior to deployment, and it is only known to the BS and the corresponding node. Further, the BS creates a temporary cluster key in each setup phase that is sent to all members of a cluster (including the cluster head). The cluster key is used by cluster members (CMs) to communicate with their cluster heads (CHs) and it is only valid in the current round. Finally, the base station (BS) generates a different “round nonce” for each WSN node that is updated once a round. An initial nonce is stored in WSN nodes prior to deployment. Each nonce is valid for one round only (the initial nonce is valid until the end of the first setup phase) and the BS keeps track of all nonce it has generated for each WSN node. Past nonce values are required to re-synchronise with nodes that may have become desynchronised due to lost messages [LDC11a] [LDC11b].

The SeBCDCP-MAC protocol has the same basic structure as the BCDCP protocol: a setup phase, which organises the clusters and routing paths, is

followed by a pre-agreed number of data communications phases. Figure 6.1 details the messages of the SeBCDCP-MAC protocol.

In step 1, all WSN nodes in the network send their IDs, location, remaining energy and a MAC produced using the key shared with the BS, and the nonce value sent by the BS in the last round. The BS, upon receiving the message, generates the MAC using the key and its own stored nonce, and compares it against the received one.

Setup Phase:	
1. $\delta \rightarrow \text{BS}$:	$ID_\delta, Loc_\delta, RE, \{ID_\delta, Loc_\delta, RE, N_{\delta-PRE}\}MAC-K_\delta$
2a. $\text{BS} \rightarrow \text{C}$:	$ID_C, \langle Path \rangle, \langle MID \rangle, N_{C-NEW}, \{K_{CC}\}K_C, \langle TSN \rangle, \{ID_C, \langle Path \rangle, \langle MID \rangle, N_{C-NEW}, \{K_{CC}\}K_C, \langle TSN \rangle, N_{C-PRE}\}MAC-K_C$
2b. $\text{BS} \rightarrow \text{C}$:	$ID_C, N_{C-PRE}, \{ID_C, N_{C-PRE}, N_{C-OLD}\}MAC-K_C$
3a. $\text{BS} \rightarrow \text{A}$:	$ID_A, ID_C, TS, N_{C-NEW}, N_{A-NEW}, \{K_{CC}\}K_A, \{ID_A, ID_C, TS, N_{C-NEW}, N_{A-NEW}, \{K_{CC}\}K_A, N_{A-PRE}\}MAC-K_A$
3b. $\text{BS} \rightarrow \text{A}$:	$ID_A, N_{A-PRE}, \{ID_A, N_{A-PRE}, N_{A-OLD}\}MAC-K_A$
Data Communication Phase:	
4. $\text{A} \rightarrow \text{C}$:	$ID_A, Data, \{ID_A, Data, increment(N_{C-NEW})\}MAC-K_{CC}$
5. $\text{C} \rightarrow \dots \rightarrow \text{BS}$:	$ID_C, Fdata, \{ID_C, Fdata, increment(N_{C-NEW})\}MAC-K_C$

Legend of additional symbols:

K_X :	Symmetric key shared by node X and BS
N_{X-PRE} :	Nonce generated by the BS for Node X in the previous round
N_{X-OLD} :	Nonce generated by the BS for node X prior to the previous round
N_{X-NEW} :	Nonce generated by the BS for node X in the current round
K_{CX} :	Cluster key shared by CH X and its CMs.
$Increment(D)$:	The function of increment value D by one
$\{DATA\}K$:	Encryption function, encrypting DATA with key K.
$\{DATA\}MAC-K$:	MAC digest computed using key K.

Figure 6.1 SeBCDCP-MAC Protocol Specification

If the two MACs match, the node is authenticated and the BS stores the node's status and waits to receive a message from all other nodes. In the case of mismatching MACs, the BS does not drop the message, but verifies the incoming MAC using the same key and all older nonce sent to the node. If a match is found, the BS assumes desynchronisation has taken place and sends the nonce an updated message (message 2b/3b in Figure 6.1). The nonce's updated message contains the receptor's ID, the nonce for the previous round, and a MAC produced by the shared key and the out-of-date nonce. BS then awaits the WSN node sending the status message again with the updated nonce. Thus, the protocol prevents Suppress-and-Desynchronise attacks [LDC11a] [LDC11b]. This node synchronization strategy should be more energy efficient than that of SecLEACH: first, SecLEACH synchronizes the nodes in each and every round. No matter whether desynchronisation happened or not, every WSN node has to consume extra energy for a key-update message. Whenever a key is received, the nodes also have to perform a number of hashing operations to verify the key. On the other hand, in SeBCDCP-MAC the BS can detect the desynchronized WSN nodes by verifying the first received message, and only synchronizes the node when necessary.

Once the BS collects the status of all WSN nodes it decides the role for each node. BS informs the CHs and CMs respectively: in step 2a, the BS sends all CHs the CH informing message. The message holds the whole content of BCDCP's message 2a (Cf. Figure 3.3), plus the following extra information for security protection:

- An updated nonce for the current round.
- The temporary cluster key, shared by the receptor CH and all its members for the current round, encrypted using the CH's private key.

- A MAC produced by the shared key and the last round's nonce.

In step 3a, the BS sends each CM its ID, the ID of its CH, the updated nonce, the time schedule for sending the collected data to the CH, and the MAC.

In the data communication phase, the node-to-CH data is protected by the MAC generated by the temporary cluster key and the CH's up-to-date nonce (step 4). The nonce is incremented by one upon sending a data message to the CH. By checking the authenticity of the messages, the CH can track missing data messages by detecting the nonce difference between two received messages. This also defends against jamming attacks, such as SD attacks [LDC11a]. When all data messages from its CMs are collected, the CH performs data fusion and sends the results to the BS in step 5. This message is protected by a MAC, using the incremented current round's nonce and the shared key.

6.3 SeBCDCP-ENC Introduction

The SeBCDCP-ENC protocol uses the same rounds as BCDP and SeBCDCP-MAC. Each round is composed of one setup phase and a pre-agreed number of data communication phases. SeBCDCP-ENC uses the same freshness mechanism as SeBCDCP-MAC by utilizing a nonce that is updated once per round. The main difference between SeBCDCP-ENC and SeBCDCP-MAC is the used security strategy: SeBCDCP-ENC adopts a block cipher to encrypt the messages, to provide authentication, confidentiality, and integrity protection. The protocol is detailed in Figure 6.2.

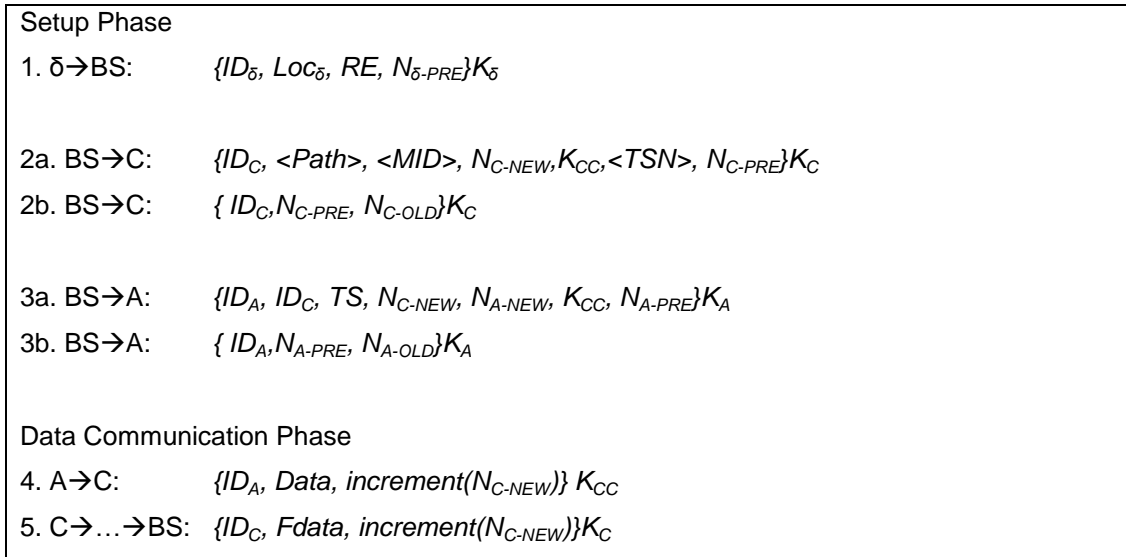


Figure 6.2 SeBCDCP-ENC Protocol Specification

In step 1, the WSN nodes send the same status information as in SeBCDCP-MAC. Additionally, the last round nonce $N_{\delta-PRE}$ is also included in the message. The message is protected by encrypting it with the key shared by the BS and the WSN node. Upon receiving the message, the BS decrypts it and matches the sender's ID and the nonce. The protocol continues analogous to SeBCDCP-MAC: if the nonce and ID match the stored values, information is passed on to the CH and CMs (steps 2a/3a). If the nonce and ID do not match a nonce updating message is sent (steps 2b/3b). While the same information as in SeBCDCP-MAC is sent, the messages are encrypted with the corresponding shared keys. As the same node synchronization strategy as SeBCDCP-MAC is adopted, SeBCDCP-ENC has the same energy efficiency advantage over SecLEACH.

The data communication phase of SeBCDCP-ENC uses the same security mechanism as in the setup phase: The CMs use the received cluster key K_{cc} to encrypt the sensed data and include the incremented nonce value in the message of step 4. The CH that receives the message decrypts and matches the received

nonce and the sender ID. When all data messages from its CMs are collected, the CH performs data fusion and sends the results together with the incremented nonce value encrypted to the BS in step 5.

6.4 Security Analysis of SeBCDCP-MAC and SeBCDCP-ENC

Both SeBCDCP-MAC and SeBCDCP-ENC aim to provide a secure clustering and routing protocol that maintains the advantages of the BCDCP protocol, yet the objectives are different: SeBCDCP-MAC suits the application with no confidentiality requirement in regards to the sensed data, where SeBCDCP-ENC provides full protection.

6.4.1 Defending Attacks Breaching Confidentiality

The SeBCDCP-MAC protocol offers cluster key confidentiality protection, yet no protection for the data (Cf. Figure 6.1 step 5 and step 6). The SeBCDCP-ENC protocol offers a full confidentiality protection to every message.

6.4.2 Defending Attacks Breaching Integrity

SeBCDCP-MAC offers integrity protection by using MAC. A malicious modification to the plaintext would not match of the attached MAC. In SeBCDCP-ENC, each message contains recognizable bits (the receptor's ID) and is encrypted in whole. Modification to the message renders the decrypted content intelligible, thus making the ID incorrect.

6.4.3 Defending Attacks Breaching Authenticity

Both SeBCDP-MAC and SeBCDCP-ENC are based on the assumption of the shared key and is only known by the legitimate parties. SeBCDCP-MAC protects the authenticity by utilizing MAC. Only the legitimate nodes can generate the same MAC digest using the shared key and the same message content. In SeBCDCP-ENC, since the key is only known to the legitimate parties, the successful decrypting of the message would prove the sender's authenticity.

6.4.4 Defending Attacks Breaching Freshness

Both SeBCDCP-ENC and SeBCDCP-MAC protect the freshness of the messages by using the nonce updating per round strategy. BS generates the fresh nonce individually for each WSN node, and updates them in every round. If the nonce used by a WSN node in message 1 of the setup phase is an out-of-date nonce, the BS will send nonce updating messages (2b and 3b in the setup phase). If a nonce in the data communication phase is outdated, then the message will be rejected. All nonce updating messages are protected by the MAC (SeBCDCP-MAC) or encryption (SeBCDCP-ENC) with the previous round's nonce.

6.5 Security Comparison of BCDP, SeBCDCP-MAC, and SeBCDCP-ENC

In SeBCDCP-MAC, each message is attached a MAC digest for protecting messages' authenticity and integrity [Inf02]. Since the nonce is involved in the MAC digest generation, the messages' freshness is protected. However, MAC algorithm cannot protect messages' confidentiality, thus the only confidential content in SeBCDCP-MAC is the cluster key. On the other hand, SeBCDCP-ENC

6. Securing BCDCP Routing Protocol

uses block cipher to protect the messages' confidentiality. By utilizing a shared key between the BS and each WSN node, plus the fresh nonce and sender / receptor's ID in specific locations attached in the message-to-be-encrypted strategy, the messages' freshness, authenticity, and integrity are protected.

In general, the security comparison of BCDCP, SeBCDCP-MAC, and SeBCDCP-ENC is shown in Table 6.1. From the table we can see, SeBCDCP-ENC provides full confidentiality, integrity, authenticity, and freshness protection to the messages; while SeBCDCP-MAC protects the messages' integrity, authenticity, and freshness, but only the cluster key's confidentiality is protected.

Table 6.1 Security Comparison of the Protocols

	BCDCP	SeBCDCP-MAC	SeBCDCP-ENC
Confidentiality	No	<i>Only Cluster Key</i>	Yes
Integrity	No	Yes	Yes
Authenticity	No	Yes	Yes
Freshness	No	Yes	Yes

7. Formal Verification of the Wireless Sensor Network Routing Protocols

In this chapter, I will formally verify the security of two WSN routing protocols, SeBCDCP-MAC and SeBCDCP-ENC, using automated logic-based CDVT tool [CoR08] [DoC05] [DLC08] [PDC08]. The aim of the formal verification is to establish if the security goals of the protocols have been proven. This gives confidence in the correctness of the design of the protocol(s).

7.1 Formal Verification using CDVT Tool

Logic-based verification involves a process of deductive reasoning, where the desired protocol goals are deduced by applying a set of axioms and inference rules to the assumptions and message exchanges of the protocol [DoC05]. CDVT verification tool is an automated system that implements a modal logic of knowledge and belief using Layered Proving Trees [DLC08]. Verification of security protocols using the CDVT Logic-based verification tool is accomplished by firstly formalizing the protocol specification in the language of the tool; a formalized protocol incorporates three components:

- (i) The protocol steps,
- (ii) The set of initial assumptions of the protocol
- (iii) The security goals of the protocol.

The formalized protocol specification is then submitted to the verification tool which uses an (automated) process of deductive reasoning to establish if the goals of protocol are proven true or false.

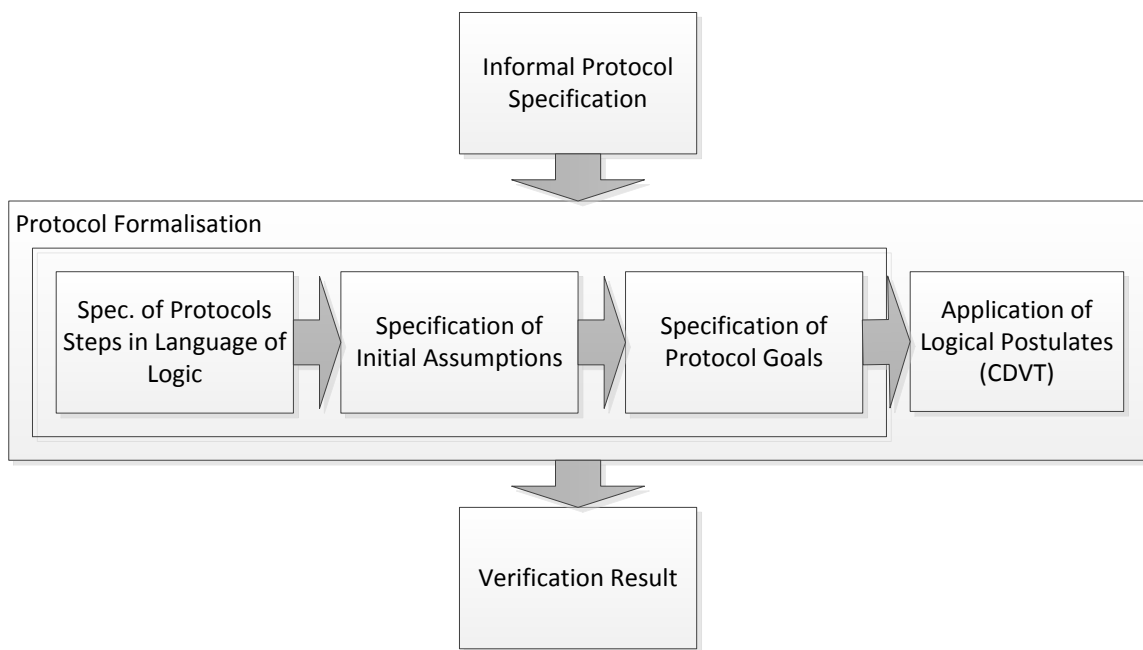


Figure 7.1 Formal Verification using CDVT tool

7.2 Formal Verification of SeBCDCP-MAC Protocol

To verify the security of SeBCDCP-MAC protocol, the receptor nodes should acknowledge the received messages' authenticity, integrity, freshness, and confidentiality of contents.

For referencing purpose, SeBCDCP-MAC's specification is listed again below:

SeBCDCP-MAC Protocol Specification

Setup Phase:	
1. A→BS:	$ID_A, Loc_A, RE, \{Loc_A, RE, N_{A-PRE}\}MAC-K_A$
2. C→BS:	$ID_C, Loc_C, RE, \{Loc_C, RE, N_{C-PRE}\}MAC-K_C$
3a. BS→C:	$ID_C, \langle Path \rangle, \langle MID \rangle, N_{C-NEW}, \{K_{CC}\}K_C, \langle TSN \rangle, \{ID_C, \langle Path \rangle, \langle MID \rangle, N_{C-NEW}, \{K_{CC}\}K_C, \langle TSN \rangle, N_{C-PRE}\}MAC-K_C$
3b. BS→C:	$ID_C, N_{C-PRE}, \{ID_C, N_{C-OLD}, N_{C-PRE}\}MAC-K_C$
4a. BS→A:	$ID_A, ID_C, TS, N_{C-NEW}, N_{A-NEW}, \{K_{CC}\}K_A, \{ID_A, ID_C, TS, N_{C-NEW}, N_{A-NEW}, \{K_{CC}\}K_A, N_{A-PRE}\}MAC-K_A$
4b. BS→A:	$ID_A, N_{A-PRE}, \{ID_A, N_{A-PRE}, N_{A-OLD}\}MAC-K_A$
Data Communication Phase:	
5. A→C:	$ID_A, Data, \{ID_A, Data, increment(N_{C-NEW})\}MAC-K_{CC}$
6. C→...→BS:	$ID_C, Fdata, \{ID_C, Fdata, increment(N_{C-NEW})\}MAC-K_C$

7.2.1 SeBCDCP-MAC Initial Assumptions

The following (A1-A19) are the formalized initial assumptions of SeBCDCP-MAC protocol:

- A1: A possess at[0] K_A ;
- A2: C possess at[0] K_C ;
- A3: BS possess at[0] K_A ;
- A4: BS possess at[0] K_C ;
- A5: BS possess at[0] K_{CC} ;
- A6: A know at[0] NOT(Zero possess at[0] N_{A-Pre});
- A7: C know at[0] NOT(Zero possess at[0] N_{C-Pre});

- A8: BS know at[0] NOT(Zero possess at[0] NcPre);
- A9: BS know at[0] NOT(Zero possess at[0] NaPre);
- A10:BS know at[0] NOT(Zero possess at[0] NaNew);
- A11:BS know at[0] NOT(Zero possess at[0] NcNew);
- A12: BS know at[0] ((BS receive at[1] A, dataLocA, Hmac_Ka(dataLocA, NaPre) AND (BS possess at[0] Ka AND BS know at[0] NOT(Zero possess at[0] NaPre))) IMPLY A send at[1] A, dataLocA, Hmac_Ka(dataLocA, NaPre));
- A13: BS know at[0] ((BS receive at[2] C, dataLocC, Hmac_Kc(dataLocC, NcPre) AND (BS possess at[0] Kc AND BS know at[0] NOT(Zero possess at[0] NcPre))) IMPLY C send at[2] C, dataLocC, Hmac_Kc(dataLocC, NcPre));
- A14: C know at[0] ((C receive at[3] C, dataPath, NcNew, {Kcc}Kc, Hmac_Kc(C, dataPath, NcNew, {Kcc}Kc, NcPre) AND (C possess at[0] Kc AND C know at[0] NOT(Zero possess at[0] NcPre))) IMPLY BS send at[3] C, dataPath, NcNew, {Kcc}Kc, Hmac_Kc(C, dataPath, NcNew, {Kcc}Kc, NcPre));
- A15: C know at[0] ((C know at[3] BS send at[3] C, dataPath, NcNew, {Kcc}Kc, Hmac_Kc(C, dataPath, NcNew, {Kcc}Kc, NcPre)) IMPLY NOT(Zero possess at[0] NcNew));
- A16: A know at[0] ((A receive at[4] A, C, dataTs, NcNew, NaNew, {Kcc}Ka, Hmac_Ka(A, C, dataTs, NcNew, NaNew, {Kcc}Ka, NaPre) AND (A possess at[0] Ka AND A know at[0] NOT(Zero possess at[0] NaPre))) IMPLY BS send at[4] A, C, dataTs, NcNew, NaNew, {Kcc}Ka, Hmac_Ka(A, C, dataTs, NcNew, NaNew, {Kcc}Ka, NaPre));
- A17: A know at[0] ((A know at[4] BS send at[4] A, C, dataTs, NcNew, NaNew, {Kcc}Ka, Hmac_Ka(A, C, dataTs, NcNew, NaNew, {Kcc}Ka, NaPre)) IMPLY NOT(Zero possess at[0] NaNew));
- A18: C know at[0] ((C receive at[5] A, dataA, Hmac_Kcc(A, dataA, NcNew) AND (C possess at[3] Kcc AND C know at[3] NOT(Zero possess at[0] NcNew))) IMPLY A send at[5] A, dataA, Hmac_Kcc(A, dataA, NcNew));
- A19: BS know at[0] ((BS receive at[6] C, dataF, Hmac_Kc(C, dataF, NcNew) AND (BS possess at[0] Kcc AND BS know at[0] NOT(Zero possess at[0] NcNew))) IMPLY C send at[6] C, dataF, Hmac_Kc(C, dataF, NcNew));

A1 expresses the facts that before the protocol starts the node A possess the key Ka, and in a similar way, A2 - A5 express the node C and the BS possesses the corresponding keys. A6 - A11 specify the knowledge of node A, C, and the BS recognizing the freshness of nonce. A12, A13, A14, A16, A18, and A19 describe

the keyed MAC rules: the received message can be verified as sent by legitimate nodes or the BS (authenticated) and was not modified during the transmission (message integrity), once the receptor principle generated MAC digest matches the received digest. To generate the same MAC digest, both sender and receptor principle have to possess the same key. If the fresh nonce recognized by both principles is involved in generating the MAC digest (not sent by plaintext), the receptor principle can verify the message's freshness. A15 and A 17 specify a trust assumption: if the received BS-sent-message containing new nonce is verified and fresh, the nonce can be trusted as the fresh nonce that the BS generated.

7.2.2 SeBCDCP-MAC Protocol Steps

The steps of the SeBCDCP-MAC protocol are formalized as follows:

- S1: BS receive at[1] A, dataLocA, Hmac_Ka(dataLocA, NaPre);
- S2: BS receive at[2] C, dataLocC, Hmac_Kc(dataLocC, NcPre);
- S3: C receive at[3] C, dataPath, NcNew, {Kcc}Kc, Hmac_Kc(C, dataPath, NcNew, {Kcc}Kc, NcPre);
- S4: A receive at[4] A, C, dataTs, NcNew, NaNew, {Kcc}Ka, Hmac_Ka(A, C, dataTs, NcNew, NaNew, {Kcc}Ka, NaPre);
- S5: C receive at[5] A, dataA, Hmac_Kcc(A, dataA, NcNew);
- S6: BS receive at[6] C, dataF, Hmac_Kc(C, dataF, NcNew);

7.2.3 SeBCDCP-MAC Protocol Goals

For formally verifying SeBCDCP-MAC, a number of goals have to be fulfilled. They are:

- G1: BS know at[1] A send at[1] A, dataLocA, Hmac_Ka(dataLocA, NaPre);
- G2: BS know at[1] NOT(Zero send at[0] A, dataLocA, Hmac_Ka(dataLocA, NaPre));
- G3: BS know at[2] C send at[2] C, dataLocC, Hmac_Kc(dataLocC, NcPre);
- G4: BS know at[2] NOT(Zero send at[0] C, dataLocC, Hmac_Kc(dataLocC, NcPre));
- G5: C know at[3] BS send at[3] C, dataPath, NcNew, {Kcc}Kc, Hmac_Kc(C, dataPath, NcNew, {Kcc}Kc, NcPre);
- G6: C know at[3] NOT(Zero send at[0] C, dataPath, NcNew, {Kcc}Kc, Hmac_Kc(C, dataPath, NcNew, {Kcc}Kc, NcPre));
- G7: C possess at[3] Kcc;
- G8: C know at[3] NOT(Zero possess at[0] NcNew);
- G9: A know at[4] BS send at[4] A, C, dataTs, NcNew, NaNew, {Kcc}Ka, Hmac_Ka(A, C, dataTs, NcNew, NaNew, {Kcc}Ka, NaPre);
- G10: A know at[4] NOT(Zero send at[0] A, C, dataTs, NcNew, NaNew, {Kcc}Ka, Hmac_Ka(A, C, dataTs, NcNew, NaNew, {Kcc}Ka, NaPre));
- G11: A possess at[4] Kcc;
- G12: A know at[4] NOT(Zero possess at[0] NaNew);
- G13: C know at[5] A send at[5] A, dataA, Hmac_Kcc(A, dataA, NcNew);
- G14: C know at[5] NOT(Zero send at[0] A, dataA, Hmac_Kcc(A, dataA, NcNew));
- G15: BS know at[6] C send at[6] C, dataF, Hmac_Kc(C, dataF, NcNew);
- G16: BS know at[6] NOT(Zero send at[0] C, dataF, Hmac_Kc(C, dataF, NcNew));

G1, G3, G5, G9, G13, and G15 are declared for the authenticity and integrity verification of each received messages. Accordingly, G2, G4, G6, G10, G14, and G16 are set for checking freshness of those messages. The goals G7 and G11 correspond to cluster key establishment: node A and C successfully possess the cluster key Kcc after Step 3 and 4. At last, G8 and G12 are used for check if the node can recognize the received nonce as fresh.

7.2.4 SeBCDCP-MAC Protocol Formal Verification Results

The formal verification results using CDVT tool is presented in Figure 7.2. The verification validates the correctness of the protocol.

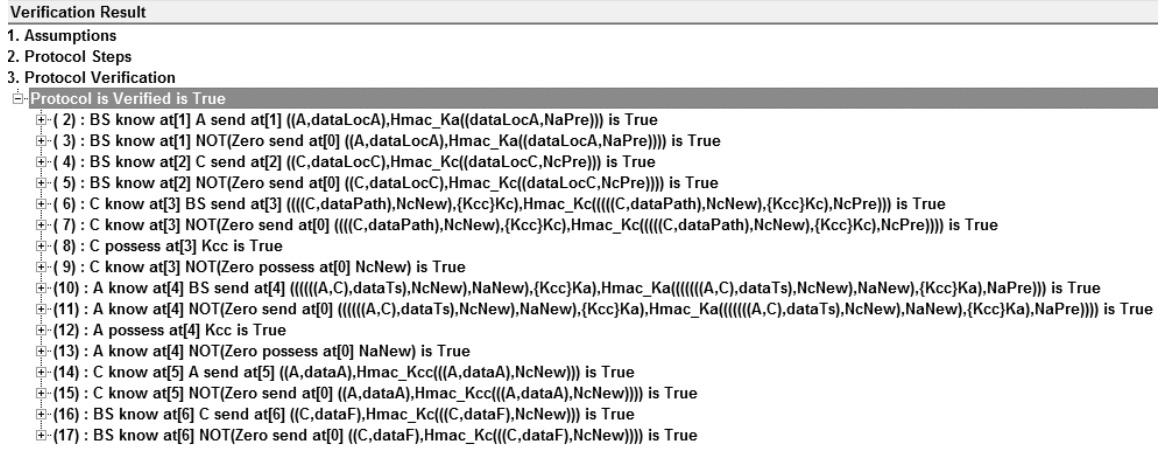


Figure 7.2 SeBCDCP-MAC Formal Verification

7.2.5 Formal Verification of the SeBCDCP-MAC Nonce Updating Steps

In SeBCDCP-MAC's specification, the two steps designed for WSN nodes' nonce synchronization (Step 2b, 3b in Figure 6.1 SeBCDCP-MAC Protocol Specification) also should be verified to guarantee containing no weakness. To verify the security of these steps, a formal verification using CDVT should be carried on.

The verification specifications are listed as follows:

Assumptions:

A1: A possess at[0] Ka;

A2: C possess at[0] Kc;

A5: A know at[0] NOT(Zero possess at[0] NaOld);

A6: C know at[0] NOT(Zero possess at[0] NcOld);

A7: C know at[0] ((C receive at[3] C, NcPre, Hmac_Kc(C, NcPre, NcOld) AND (C possess at[0] Kc AND C know at[0] NOT(Zero possess at[0] NcOld))) IMPLY BS send at[3] C, NcPre, Hmac_Kc(C, NcPre, NcOld));

A8: C know at[0] ((C know at[3] BS send at[3] C, NcPre, Hmac_Kc(C, NcPre, NcOld)) IMPLY NOT(Zero possess at[0] NcPre));

A9: A know at[0] ((A receive at[4] A, NaPre, Hmac_Ka(A, NaPre, NaOld) AND (A possess at[0] Ka AND A know at[0] NOT(Zero possess at[0] NaOld))) IMPLY BS send at[4] A, NaPre, Hmac_Ka(A, NaPre, NaOld));

A10: A know at[0] ((A know at[4] BS send at[4] A, NaPre, Hmac_Ka(A, NaPre, NaOld)) IMPLY NOT(Zero possess at[0] NaPre));

Steps:

S1: BS receive at[1] A, dataLocA, Hmac_Ka(dataLocA, NaOld);

S2: BS receive at[2] C, dataLocC, Hmac_Kc(dataLocC, NcOld);

S3: C receive at[3] C, NcPre, Hmac_Kc(C, NcPre, NcOld);

S4: A receive at[4] A, NaPre, Hmac_Ka(A, NaPre, NaOld);

Goals:

G1: C know at[3] BS send at[3] C, NcPre, Hmac_Kc(C, NcPre, NcOld);

G2: C know at[3] NOT (Zero send at[0] C, NcPre, Hmac_Kc(C, NcPre, NcOld));

G3: C know at[3] NOT (Zero possess at[0] NcPre);

G4: A know at[4] BS send at[4] A, NaPre, Hmac_Ka(A, NaPre, NaOld);

G5: A know at[4] NOT (Zero send at[0] A, NaPre, Hmac_Ka(A, NaPre, NaOld));

G6: A know at[4] NOT (Zero possess at[0] NaPre);

The purpose of the step 2b and 3b is updating the nonce for the legitimated WSN nodes. The goals are designed for verifying if it is achieved. Figure 7.3 details the CDVT verification results of the goals listed above. The verification results shows the messages are secured in the logic perspective.

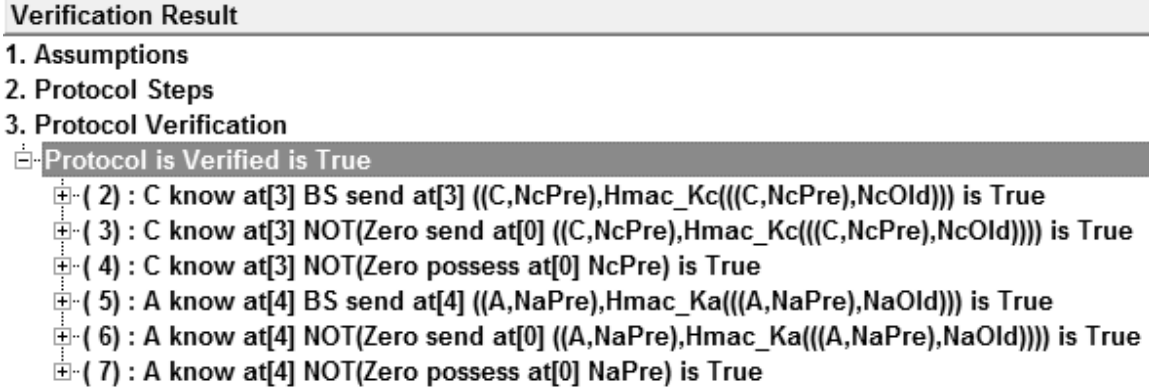


Figure 7.3 SeBCDCP-MAC Step 2b, 3b Formal Verification Results

7.3 Formal Verification of SeBCDCP-ENC Protocol

SeBCDCP-ENC has similar security goals as SeBCDCP-MAC to be fulfilled: both SeBCDCP-MAC and SeBCDCP-ENC should protect the messages' authentication, integrity, and freshness. Furthermore, SeBCDCP-ENC shall provide full confidentiality protection to all messages. Again, the SeBCDCP-ENC's protocol specification is re-listed for the referencing purpose.

SeBCDCP-ENC Protocol Specification

Setup Phase	
1. $\delta \rightarrow BS$:	$\{ID_{\delta}, Loc_{\delta}, RE, N_{\delta-PRE}\}K_{\delta}$
2. $\delta \rightarrow BS$:	$\{ID_{\delta}, Loc_{\delta}, RE, N_{\delta-PRE}\}K_{\delta}$
3a. $BS \rightarrow C$:	$\{ID_C, \langle Path \rangle, \langle MID \rangle, N_{C-NEW}, K_{CC}, \langle TSN \rangle, N_{C-PRE}\}K_C$
3b. $BS \rightarrow C$:	$\{ID_C, N_{C-PRE}, N_{C-OLD}\}K_C$
4a. $BS \rightarrow A$:	$\{ID_A, ID_C, TS, N_{C-NEW}, N_{A-NEW}, K_{CC}, N_{A-PRE}\}K_A$
4b. $BS \rightarrow A$:	$\{ID_A, N_{A-PRE}, N_{A-OLD}\}K_A$
Data Communication Phase	
5. $A \rightarrow C$:	$\{ID_A, Data, increment(N_{C-NEW})\}K_{CC}$
6. $C \rightarrow \dots \rightarrow BS$:	$\{ID_C, Fdata, increment(N_{C-NEW})\}K_C$

7.3.1 SeBCDCP-ENC Initial Assumptions

The following specifications are the formal initial assumptions of SeBCDCP-ENC protocol.

- A1: A possess at[0] Ka;
- A2: C possess at[0] Kc;
- A3: BS possess at[0] Ka;
- A4: BS possess at[0] Kc;
- A5: A know at[0] BS possess at[0] Ka;
- A6: C know at[0] BS possess at[0] Kc;
- A7: BS know at[0] A possess at[0] Ka;
- A8: BS know at[0] C possess at[0] Kc;
- A9: BS know at[0] NOT(Zero possess at[0] NcPre);
- A10: BS know at[0] NOT(Zero possess at[0] NcNew);
- A11: BS know at[0] NOT(Zero possess at[0] NaPre);
- A12: BS know at[0] NOT(Zero possess at[0] NaNew);
- A13: A know at[0] NOT(Zero possess at[0] NaPre);
- A14: C know at[0] NOT(Zero possess at[0] NcPre);
- A15: C know at[0] ((C know at[3] BS send at[3] {C, dataPath, NcNew, Kcc, NcPre}Kc) IMPLY NOT(Zero possess at[0] NcNew));
- A16: C know at[0] ((C possess at[3] Kcc AND C possess at[3] dataPath) IMPLY A possess at[5] Kcc);
- A17: C know at[0] ((C possess at[5] Kcc AND C possess at[5] dataPath) IMPLY NOT(Zero possess at[0] NcNew));
- A18: A know at[0] ((A know at[4] BS send at[4] {A, C, dataTs, NcNew, Kcc, NaNew, NaPre}Ka) IMPLY NOT(Zero possess at[0] NaNew));

SeBCDCP-ENC has similar assumptions with SeBCDCP-MAC. The differences are: A5 - A8 indicates the nodes know that the BS shares same keys with them, and vice versa. A16 expresses the protocol's assumption: if C successfully verifies

the 3rd step message, it knows the node A is its member and will obtain the same cluster key Kcc at step 4.

7.3.2 SeBCDCP-ENC Protocol Steps

The steps of the SeBCDCP-MAC protocol are formalized as follows:

- S1: BS receive at[1] {A, dataLocA, NaPre}Ka;
- S2: BS receive at[2] {C, dataLocC, NcPre}Kc;
- S3: C receive at[3] {C, dataPath, NcNew, Kcc, NcPre}Kc;
- S4: A receive at[4] {A, C, dataTs, NcNew, Kcc, NaNew, NaPre}Ka;
- S5: C receive at[5] {A, dataA, NcNew}Kcc;
- S6: BS receive at[6] {C, dataF, NcNew}Kc;

7.3.3 SeBCDCP-MAC Protocol Goals

The main objective of the goals are verifying if the receptor nodes or the BS successfully obtain the expected content, and if the messages' authenticity, integrity, and freshness are protected. The security goals to be fulfilled are:

- G1: BS know at[1] A send at[1] {A, dataLocA, NaPre}Ka;
- G2: BS know at[1] NOT(Zero send at[0] {A, dataLocA, NaPre}Ka);
- G3: BS know at[2] C send at[2] {C, dataLocC, NcPre}Kc;
- G4: BS know at[2] NOT(Zero send at[0] {C, dataLocC, NcPre}Kc);
- G5: C possess at[3] Kcc;
- G6: C know at[3] BS send at[3] {C, dataPath, NcNew, Kcc, NcPre}Kc;
- G7: C know at[3] NOT(Zero send at[0] {C, dataPath, NcNew, Kcc, NcPre}Kc);
- G8: C know at[3] NOT(Zero possess at[0] NcNew);
- G9: A know at[4] NOT(Zero possess at[0] NaNew);

- G10: A possess at[4] Kcc;
 G11: A know at[4] BS send at[4] {A, C, dataTs, NcNew, Kcc, NaNew, NaPre}Ka;
 G12: A know at[4] NOT(Zero send at[4] {A, C, dataTs, NcNew, Kcc, NaNew, NaPre}Ka);
 G13: C know at[5] A send at[5] {A, dataA, NcNew}Kcc;
 G14: C know at[5] NOT(Zero send at[0] {A, dataA, NcNew}Kcc);
 G15: BS know at[6] C send at[6] {C, dataF, NcNew}Kc;
 G16: BS know at[6] NOT(Zero send at[0] {C, dataF, NcNew}Kc);

7.3.4 SeBCDCP-ENC Protocol Formal Verification Results

The CDVT formal verification results' screenshot is in Figure 7.4. The results reveal SeBCDCP-ENC is secure in logical perspective.

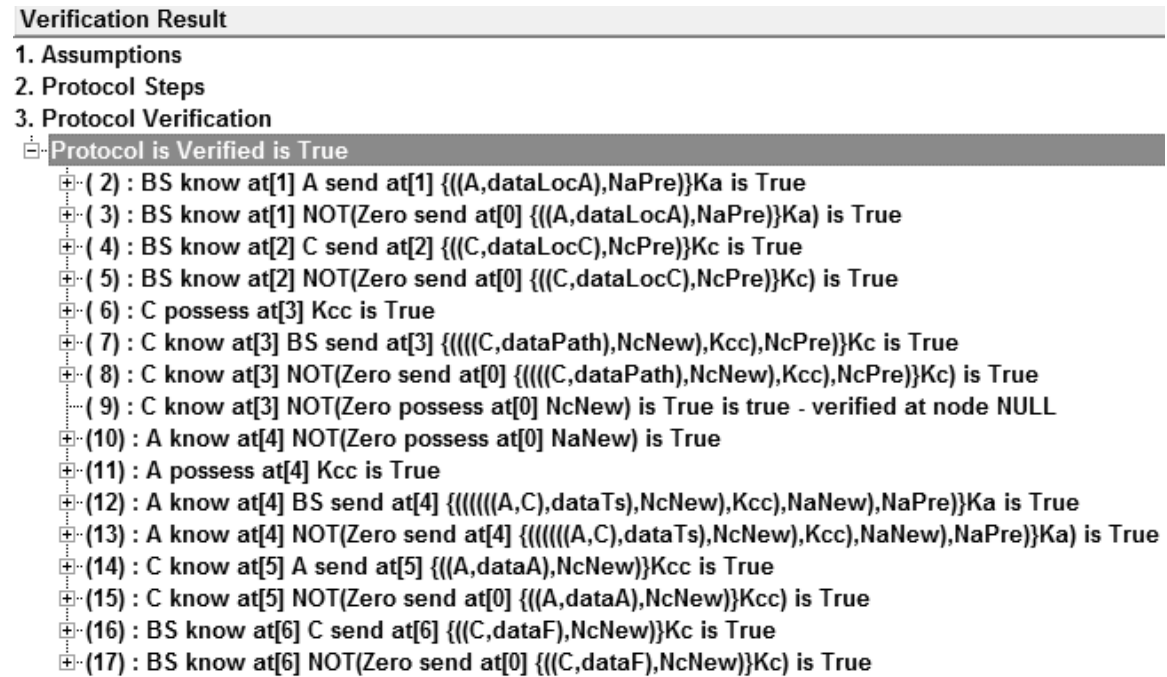


Figure 7.4 SeBCDCP-ENC Formal Verification

7.3.5 Formal Verification of the SeBCDCP-ENC Nonce Updating Steps

With same reasoning as for SeBCDCP-MAC, I will formally verify the security of step 2b and 3b of SeBCDCP-ENC (Cf. Figure 6.2). The formal specifications are listed as follows:

Assumptions:

A1: A possess at[0] Ka;

A2: C possess at[0] Kc;

A3: A know at[0] BS possess at[0] Ka;

A4: C know at[0] BS possess at[0] Kc;

A5: A know at[0] NOT(Zero possess at[0] NaOld);

A6: C know at[0] NOT(Zero possess at[0] NcOld);

A7: C know at[0] ((C know at[3] BS send at[3] {C, NcPre, NcOld}Kc) IMPLY NOT(Zero possess at[0] NcPre));

A8: A know at[0] ((A know at[4] BS send at[4] {A, NaPre, NaOld}Ka) IMPLY NOT(Zero possess at[0] NaPre));

Steps:

S1: BS receive at[1] {A, dataLocA, NaOld}Ka;

S2: BS receive at[2] {C, dataLocC, NcOld}Kc;

S3: C receive at[3] {C, NcPre, NcOld}Kc;

S4: A receive at[3] {A, NaPre, NaOld}Ka;

Goals:

G1: C know at[3] BS send at[3] {C, NcPre, NcOld}Kc;

- G2: C know at[3] NOT(Zero send at[0] {C, NcPre, NcOld}Kc);
 G3: C know at[3] NOT(Zero possess at[0] NcPre);
 G4: A know at[4] BS send at[4] {A, NaPre, NaOld}Ka;
 G5: A know at[4] NOT(Zero send at[0] {A, NaPre, NaOld}Ka);
 G6: A know at[4] NOT(Zero possess at[0] NaPre);

The CDVT formal verification results are listed in Figure 7.5. As the results show, the protocol is verified to be secure in the logical perspective.

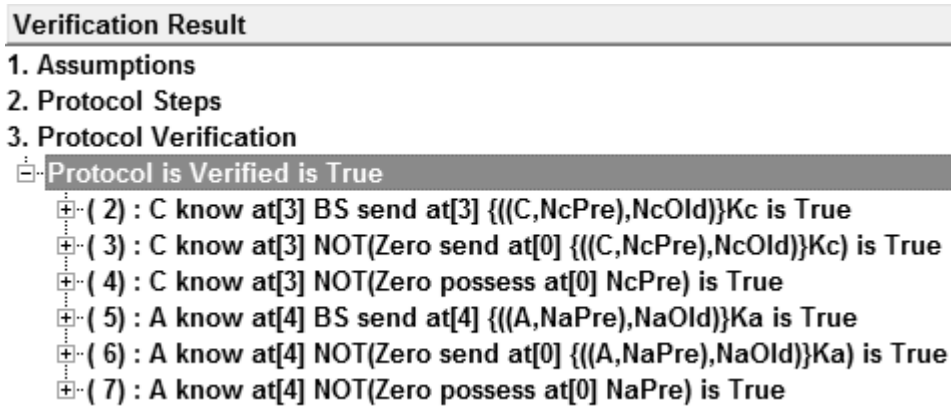


Figure 7.5 SeBCDCP-ENC Step 2b, 3b Formal Verification Results

This chapter formally verifies the security of two WSN routing protocols, SeBCDCP-MAC and SeBCDCP-ENC. Both protocols are verified to successfully provide authenticity, integrity, and freshness protections to the messages. SeBCDCP-MAC protects only the cluster key's confidentiality, but SeBCDCP-ENC provides full confidentiality protection to all messages.

8. Theoretical Estimation of the Energy Consumption of Wireless Sensor Network Routing Protocols Using Heinzelman's Model

In [MMB05], the authors have used Heinzelman's energy consumption estimation model [HCB02] to evaluate BCDCP's energy consumption. In this chapter, I am going to use the same model to estimate the energy consumption of three WSN routing protocols: BCDCP, SeBCDCP-MAC, and SeBCDCP.

The key variables in Heinzelman's model are:

- The amount of bytes to be transmitted/received
- The communication distance.

It is assumed the three protocols' implementations have the same experimental conditions, as in the same amount of homogeneous sensor hardware located in the same geographic positions in the network. Thus, the only energy consumption difference caused should be the transmitted and received byte amount difference.

8.1 Calculating Total Number of Transmit-Received Bytes

According to the radio model discussed in [HCB02] and [MMB05], the main factors in deciding the energy consumption are the amount of bytes to be transmitted and received, and the transmission distance. The longer the transmission distance and the larger the message size, the more energy is required. Under the assumption that the protocols are utilized in equivalent networks, i.e. in networks using the same node hardware and having the same distance between any two WSN nodes, the message size becomes the only relevant factor when comparing the energy dissipation of the three protocol variants. Therefore, this section provides equations to calculate the number of bytes transmitted/received by individual nodes. Further, the presented analysis distinguishes between simple CMs and CH, as the corresponding message sizes vary significantly. Due to the radio transceivers' very similar energy consumption level between the transmission mode and reception mode for some WSN node hardware ([Cro12a] and [Cro10b]), the analysis also assumes the energy consumed for transmitting and receiving a byte are the same.

Table 8.1 summarises the used notation. The total bytes per round (M) is subscripted to indicate the protocol, the network role, the communication activity and the scenario as outlined in Table 8.2.

Table 8.1 Notation for Message Size Computations

Symbol	Description
$M_{\text{subscript}}$	Total bytes per round
B_x	Message size in bytes of x-th step in original BCDCP protocol
N_{DataCom}	Number of data communication phases per round
N_{Cluster}	Number of other clusters
N_{Member}	Number of nodes in a cluster
S_{MAC}	MAC digest size in bytes
S_{ENC}	Encryption block size in bytes
S_{Key}	Key size in bytes
S_{Nonce}	Nonce size in bytes
$F(x)$	Size of ciphertext of x – always an integer multiple of S_{ENC}

Table 8.2 Total Bytes per Round Subscripts

Subscript	Protocol
BCDCP	Original BCDCP protocol
SBMAC	SeBCDCP-MAC Protocol
SBENC	SeBCDCP-ENC Protocol
	Network Role
CH	Cluster Head
CM	Cluster Member
	Communication Activity
T	Transmission
R	Reception
	Scenario
LST	CH at end of CH Routing Path
FST	First CH in CH Routing Path

➤ *BCDCP Bytes Estimation*

For BCDCP, the per-round total transmitted bytes of a CM are calculated by Eq. 8.1.

$$M_{BCDCP-CM-T} = B_1 + N_{DataCom} * B_4 \quad (\text{Eq. 8.1})$$

In each round a CM transmits once message 1 (B_1), and for each communication phase it transmits message 4 (B_4). A CM only receives message 3 (B_3) once per round, and the total number of received bytes is calculated by Eq. 8.2.

$$M_{BCDCP-CM-R} = B_3 \quad (\text{Eq. 8.2})$$

The total number of bytes processed by the radio transmitter for a CM in a round is presented in Eq. 8.3.

$$M_{BCDCP-CM} = M_{BCDCP-CM-T} + M_{BCDCP-CM-R} = B_1 + B_3 + N_{DataCom} * B_4 \quad (\text{Eq. 8.3})$$

Analysis of the size of CH messages is more complex, as CH may need to route fusion data messages from other CHs. Once a CH is located at the end of CH-to-

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

CH routing path (LST), it does not need to forward any data messages and the number of bytes transmitted by a CH per round are calculated by Eq. 8.4, and the number of bytes received by the CH is calculated by Eq. 8.5:

$$M_{\text{BCDCP-CH-T-LST}} = B_1 + N_{\text{DataCom}} * B_5 \quad (\text{Eq. 8.4})$$

$$M_{\text{BCDCP-CH-R-LST}} = B_2 + N_{\text{DataCom}} * N_{\text{Member}} * B_4 \quad (\text{Eq. 8.5})$$

In Eq. 8.4, a CH transmits once message 1 (B_1) and in each communication phase it transmits the aggregated data from its cluster in message 5 (B_5). In Eq. 8.5, in each round a CH will once receive routing and cluster membership information and time scheduling information in message 2 (B_2). Further, in each communication phase a CH receives message 4 (B_4) from each CM.

Thus, the total transmit-received bytes for an end-of-path-CH in a round are calculated in Eq. 8.6.

$$\begin{aligned} M_{\text{BCDCP-CH-LST}} &= M_{\text{BCDCP-CH-T-LST}} + M_{\text{BCDCP-CH-R-LST}} \\ &= B_1 + B_2 + N_{\text{DataCom}} * (N_{\text{Member}} * B_4 + B_5) \end{aligned} \quad (\text{Eq. 8.6})$$

If a CH is positioned at the first in the CH-to-CH routing path (FST), it needs to route fusion data messages for all other CHs and the total number of bytes transmitted and received by a CH per round are calculated by Eq. 8.7 and Eq. 8.8, respectively:

$$M_{\text{BCDCP-CH-T-FST}} = M_{\text{BCDCP-CH-T-LST}} + N_{\text{Cluster}} * N_{\text{DataCom}} * B_5 \quad (\text{Eq. 8.7})$$

$$M_{\text{BCDCP-CH-R-FST}} = M_{\text{BCDCP-CH-R-LST}} + N_{\text{Cluster}} * N_{\text{DataCom}} * B_5 \quad (\text{Eq. 8.8})$$

In addition to its own aggregated data as per Eq. 8.4 and Eq. 8.5, it also needs to transmit the aggregated data of every other CH. The overall per round transmit-

received bytes for a CH in the first phase of the CH-to-CH routing path are calculated in Eq. 8.9.

$$\begin{aligned} M_{\text{BCDCP-CH-FST}} &= M_{\text{BCDCP-CH-T-FST}} + M_{\text{BCDCP-CH-R-FST}} \\ &= M_{\text{BCDCP-CH-LST}} + 2 * N_{\text{Cluster}} * N_{\text{DataCom}} * B_5 \end{aligned} \quad (\text{Eq. 8.9})$$

➤ *SeBCDCP-MAC Bytes Estimation*

The following performance analysis of SeBCDCP-MAC assumes that no message desynchronisation takes place. Consequently, messages 2b and 3b (Cf. Figure 6.1) are not considered in the analysis. Compared to BCDCP, the number of bytes transmitted by a CM in BCDCP-MAC is increased by the size of a MAC digest for each message as shown in Eq. 8.10.

$$M_{\text{SBMAC-CM-T}} = B_1 + S_{\text{Mac}} + N_{\text{DataCom}} * (S_{\text{Mac}} + B_4) \quad (\text{Eq. 8.10})$$

The number of bytes received is increased by the size of a nonce, the size of an encrypted key and the size of MAC digest as shown in Eq. 8.11. The total transmit-received bytes for a CM in a round are presented in Eq. 8.12.

$$M_{\text{SBMAC-CM-R}} = B_3 + 2 * S_{\text{Nonce}} + F(S_{\text{Key}}) + S_{\text{Mac}} \quad (\text{Eq. 8.11})$$

$$\begin{aligned} M_{\text{SBMAC-CM}} &= M_{\text{SBMAC-CM-T}} + M_{\text{SBMAC-CM-R}} = B_1 + B_3 + 2 * S_{\text{Nonce}} + 2 * S_{\text{Mac}} + F(S_{\text{Key}}) + \\ &N_{\text{DataCom}} * (S_{\text{Mac}} + B_4) \end{aligned} \quad (\text{Eq. 8.12})$$

The number of bytes transmitted and received by CHs in the first and last positions in the routing path scenarios are increased accordingly, as expressed by Eq. 8.13 – Eq. 8.18.

$$M_{\text{SBMAC-CH-T-LST}} = B_1 + S_{\text{Mac}} + N_{\text{DataCom}} * (S_{\text{Mac}} + B_5) \quad (\text{Eq. 8.13})$$

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

$$M_{\text{SBMAC-CH-R-LST}} = B_2 + S_{\text{Nonce}} + F(S_{\text{Key}}) + S_{\text{Mac}} + N_{\text{DataCom}} * N_{\text{Member}} * (S_{\text{Mac}} + B_4) \quad (\text{Eq. 8.14})$$

$$M_{\text{SBMAC-CH-LST}} = M_{\text{SBMAC-CH-T-LST}} + M_{\text{SBMAC-CH-R-LST}} = B_1 + B_2 + 2 * S_{\text{Mac}} + S_{\text{Nonce}} + F(S_{\text{Key}}) + N_{\text{DataCom}} * ((S_{\text{Mac}} + B_5) + N_{\text{Member}} * (S_{\text{Mac}} + B_4)) \quad (\text{Eq. 8.15})$$

$$M_{\text{SBMAC-CH-T-FST}} = M_{\text{SBMAC-CH-T-LST}} + N_{\text{Cluster}} * N_{\text{DataCom}} * (S_{\text{Mac}} + B_5) \quad (\text{Eq. 8.16})$$

$$M_{\text{SBMAC-CH-R-FST}} = M_{\text{SBMAC-CH-R-LST}} + N_{\text{Cluster}} * N_{\text{DataCom}} * (S_{\text{Mac}} + B_5) \quad (\text{Eq. 8.17})$$

$$M_{\text{SBMAC-CH-FST}} = M_{\text{SBMAC-CH-T-FST}} + M_{\text{SBMAC-CH-R-LST}} = M_{\text{SBMAC-CH-LST}} + 2 * N_{\text{Cluster}} * N_{\text{DataCom}} * (S_{\text{Mac}} + B_5) \quad (\text{Eq. 8.18})$$

➤ *SeBCDCP-ENC Bytes Estimation*

Same as SeBCDCP-MAC's byte amount calculation, the following performance analysis of SeBCDCP-ENC assumes that no message desynchronisation takes place. Consequently, messages 2b and 3b are not considered in the analysis. The number of bytes transmitted by a CM is calculated by Eq. 8.19:

$$M_{\text{SBENC-CM-T}} = F(B_1 + S_{\text{Nonce}}) + N_{\text{DataCom}} * F(S_{\text{Nonce}} + B_4) \quad (\text{Eq. 8.19})$$

In each round a CM transmits the cipher block of message 1 ($F(B_1 + S_{\text{Nonce}})$) and for each data communication phase the cipher block of message 4 ($F(B_4 + S_{\text{Nonce}})$).

A cluster member receives only message 3 – the number of bytes is calculated by Eq. 8.20. The total transmit-received bytes for a CM in a round are presented in Eq. 8.21.

$$M_{\text{SBENC-CM-R}} = F(B_3 + 3 * S_{\text{Nonce}} + S_{\text{Key}}) \quad (\text{Eq. 8.20})$$

$$M_{\text{SBENC-CM}} = F(B_1 + S_{\text{Nonce}}) + N_{\text{DataCom}} * F(S_{\text{Nonce}} + B_4) + F(B_3 + 3 * S_{\text{Nonce}} + S_{\text{Key}}) \quad (\text{Eq. 8.21})$$

Correspondingly, the number of bytes transmitted and received by CHs takes into account the cipher block size and the increased messages sizes due to additional nonces and keys in the messages as shown in Eq. 8.22 – Eq. 8.27.

$$M_{\text{SBENC-CH-T-LST}} = F(B_1 + S_{\text{Nonce}}) + N_{\text{DataCom}} * F(S_{\text{Nonce}} + B_5) \quad (\text{Eq. 8.22})$$

$$M_{\text{SBENC-CH-R-LST}} = F(B_2 + 2 * S_{\text{Nonce}} + S_{\text{Key}}) + N_{\text{DataCom}} * N_{\text{Member}} * F(S_{\text{Nonce}} + B_4) \quad (\text{Eq. 8.23})$$

$$\begin{aligned} M_{\text{SBENC-CH-LST}} &= M_{\text{SBENC-CH-T-LST}} + M_{\text{SBENC-CH-R-LST}} \\ &= F(B_1 + S_{\text{Nonce}}) + F(B_2 + 2 * S_{\text{Nonce}} + S_{\text{Key}}) + N_{\text{DataCom}} * (F(S_{\text{Nonce}} + B_5) \\ &\quad + N_{\text{Member}} * F(S_{\text{Nonce}} + B_4)) \end{aligned} \quad (\text{Eq. 8.24})$$

$$M_{\text{SBENC-CH-T-FST}} = M_{\text{SBENC-CH-T-LST}} + N_{\text{Cluster}} * N_{\text{DataCom}} * F(S_{\text{Nonce}} + B_5) \quad (\text{Eq. 8.25})$$

$$M_{\text{SBENC-CH-R-FST}} = M_{\text{SBENC-CH-R-LST}} + N_{\text{Cluster}} * N_{\text{DataCom}} * F(S_{\text{Nonce}} + B_5) \quad (\text{Eq. 8.26})$$

$$\begin{aligned} M_{\text{SBENC-CH-WST}} &= M_{\text{SBENC-CH-T-FST}} + M_{\text{SBENC-CH-R-FST}} \\ &= M_{\text{SBENC-CH-LST}} + 2 * N_{\text{Cluster}} * N_{\text{DataCom}} * F(S_{\text{Nonce}} + B_5) \end{aligned} \quad (\text{Eq. 8.27})$$

8.2 Sample Byte Amount Computations

In the following performance comparisons, the same metrics for BCDCP as used by [MMB05] are assumed: BCDCP's step 1, 2, and 3 have a fixed size of 50 bytes, where step 4 has 100 bytes and step 5 has 500 bytes. In line with [OFV07], a nonce size of 4 bytes and a MAC digest fragment of 8 bytes are assumed. The key and block cipher length are assumed to be 16 bytes (in line with 128bit AES [DaR99]).

To compute the theoretically overall transmission and reception bytes for both the CMs and the CHs running the three protocols, we further assume an experimental WSN with 10 clusters; each cluster contains 1 CH and 50 members. All three protocols have the same 1 setup phase and 100 data communication

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

phases in a round. To better describe how the protocols message overheads are calculated, the computation procedure of BCDCP's total transmission and reception bytes is detailed here:

$$M_{\text{BCDCP-CM-T}} = 50 + 100 * 100 = 10050 \text{ bytes}$$

$$M_{\text{BCDCP-CM-R}} = 50 \text{ bytes}$$

$$M_{\text{BCDCP-CM}} = 10050 + 50 = 10100 \text{ bytes}$$

$$M_{\text{BCDCP-CH-T-LST}} = 50 + 100 * 500 = 50050 \text{ bytes}$$

$$M_{\text{BCDCP-CH-R-LST}} = 50 + 100 * 50 * 100 = 500050 \text{ bytes}$$

$$M_{\text{BCDCP-CH-LST}} = 50050 + 500050 = 550100 \text{ bytes}$$

$$M_{\text{BCDCP-CH-T-FST}} = 50050 + 9 * 100 * 500 = 500050 \text{ bytes}$$

$$M_{\text{BCDCP-CH-R-FST}} = 500050 + 9 * 100 * 500 = 950050 \text{ bytes}$$

$$M_{\text{BCDCP-CH-FST}} = 500050 + 950050 = 1450100 \text{ bytes}$$

Analogous, SeBCDCP-MAC and SeBCDCP-ENC's total transmission and reception bytes are computed as follows:

$$M_{\text{SBMAC-CM-T}} = 50 + 8 + 100 * (8 + 100) = 10858 \text{ bytes}$$

$$M_{\text{SBMAC-CM-R}} = 50 + 2 * 4 + F(16) + 8 = 82 \text{ bytes}$$

$$M_{\text{SBMAC-CM}} = 10858 + 82 = 10940 \text{ bytes}$$

$$M_{\text{SBMAC-CH-T-LST}} = 50 + 8 + 100 * (8 + 500) = 50858 \text{ bytes}$$

$$M_{\text{SBMAC-CH-R-LST}} = 50 + 8 + 4 + F(16) + 100 * 50 * (8 + 100) = 540078 \text{ bytes}$$

$$M_{\text{SBMAC-CH-LST}} = 50858 + 540078 = 590936 \text{ bytes}$$

$$M_{\text{SBMAC-CH-T-FST}} = 50858 + 9 * 100 * (8 + 500) = 508058 \text{ bytes}$$

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

$$M_{SBMAC-CH-R-FST} = 540078 + 9 * 100 * (8 + 500) = 997278 \text{ bytes}$$

$$M_{SBMAC-CH-FST} = 508058 + 997278 = 1505336 \text{ bytes}$$

$$M_{SBENC-CM-T} = F(50 + 4) + 100 * F(4 + 100) = 11264 \text{ bytes}$$

$$M_{SBENC-CM-R} = F(50 + 12 + 16) = 80 \text{ bytes}$$

$$M_{SBENC-CM} = 11264 + 80 = 11344 \text{ bytes}$$

$$M_{SBENC-CH-T-LST} = F(50 + 4) + 100 * F(500 + 4) = 51264 \text{ bytes}$$

$$M_{SBENC-CH-R-LST} = F(50 + 8 + 16) + 100 * 50 * F(100 + 4) = 560080 \text{ bytes}$$

$$M_{SBENC-CH-LST} = 51264 + 560080 = 611344 \text{ bytes}$$

$$M_{SBENC-CH-T-FST} = 51264 + 9 * 100 * F(4 + 500) = 512064 \text{ bytes}$$

$$M_{SBENC-CH-R-FST} = 560080 + 9 * 100 * F(4 + 500) = 1020880 \text{ bytes}$$

$$M_{SBENC-CH-FST} = 512064 + 1020880 = 1532944 \text{ bytes}$$

Table 8.3 Sample Computations

	BCDCP	Ratio	SBMAC	Ratio	SBENC	Ratio
CM-T	10050	100%	10858	108.04%	11264	112.08%
CM-R	50	100%	82	164%	80	160%
CH-T-LST	50050	100%	50858	101.61%	51264	102.43%
CH-T-FST	500050	100%	508058	101.60%	512064	102.40%
CH-R-LST	500050	100%	540078	108.00%	560080	112.01%
CH-R-FST	950050	100%	997278	104.97%	1020880	107.46
Total-CM	10100	100%	10940	108.31	11344	12.31%
Total-CH-LST	550100	100%	590936	107.42%	611344	111.13%
Total-CH-WST	1450100	100%	1505336	103.81%	1532944	105.71%

Table 8.3 summarises the computational results for this setup. As can be seen, SeBCDCP-MAC and SeBCDCP-ENC have close performance on the extra energy consumption on the radio transceiver. In most cases the additional message overhead is well below 10%. The most significant exception is the number of

bytes received by a CM, which increases by about 60%. However, as the number of bytes received by a CM is very small in comparison to other messages, this has very little effect on the overall message overhead.

8.3 Estimation Performance Comparison of BCDCP, SeBCDCP-MAC, and SeBCDCP-ENC

The following section compares the performance of the three protocols under various network configurations. To assess the performance of SeBCDCP-MAC and SeBCDCP-ENC, we compare their messages overheads against the original BCDCP. Due to the possibly large message size difference, the CM and the CH are differentiated again in the comparison.

➤ *Varied Number of Per Round Data Communication Phases Performance Comparison*

All BCDCP variants work in rounds: a setup phase is followed by a pre-agreed number of data communications phases (N_{DataCom}). Figure 8.1 depicts the relative message overhead of CM nodes of SeBCDCP-MAC and SeBCDCP-ENC, compared to the original BCDCP protocol in regards to the in dependency of the number of communication phases per round. Except for the number of data communication phases (N_{DataCom}), all assumptions as discussed in Section 8.2 are adopted in Figure 8.1.

As can be seen, if there is only one data communication phase per setup phase, CMs of SeBCDCP-MAC have a message overhead of 124.0% and CMs of SeBCDCP-ENC have a message overhead of 128.0%.

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

Secured protocols' CMs have minimum extra message overhead in the data communication phase. For SeBCDCP-MAC, the message overheads are the original BCDCP's 4th step message length plus the MAC digest size (108 bytes). For SeBCDCP-ENC, the overheads are 112 bytes. The original BCDCP's 4th message has the size of 100 bytes. Thus, as the number of $N_{DataCom}$ increases, the SeBCDCP-MAC's relative overhead drops to 108.3% and will infinitely close to 108%. For SeBCDCP-ENC, this value is 112.3% when $N_{DataCom}$ reaches 100 and will eventually infinitely close to 112%.

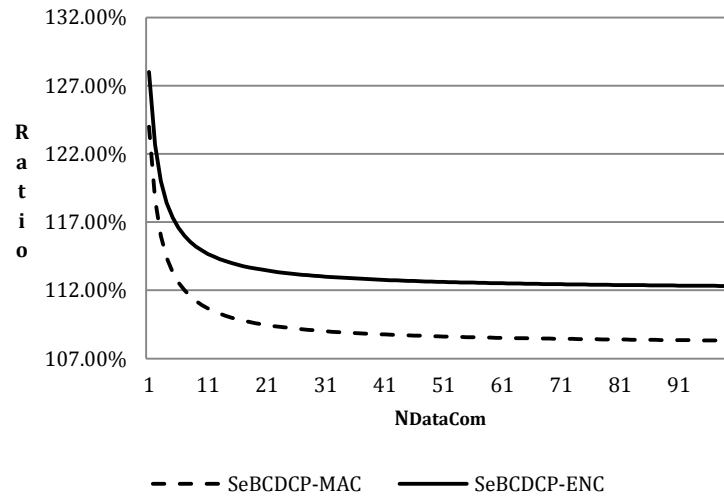


Figure 8.1 CM Relative Message Overhead in dependency of $N_{DataCom}$

Analysis of CH message overhead is more complex, as it also depends on the number of member nodes per cluster, the total number of clusters and the location of the CH in the CH-to-CH routing path. In the following analysis we assume 50 member nodes per cluster and a total of 10 clusters.

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

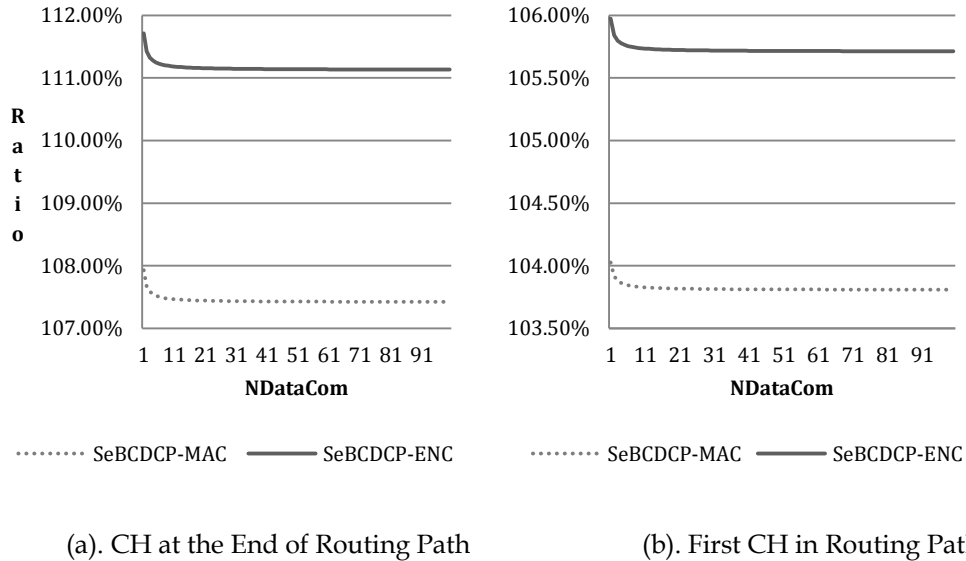


Figure 8.2 Ratio of Protocols' CH Message Overhead, in Relation with $N_{DataCom}$

The first CH on the CH-to-CH routing path only needs to deal with data from its own cluster, while CH further down the path additionally needs to forward data from previous CHs. Figure 8.2 (a) shows the relative message overhead for a first CH on a CH-to-CH routing path in dependency of $N_{DataCom}$. For a single data communication phase a SeBCDCP-MAC CH has an overhead of 107.9% and a SeBCDCP-ENC CH has an overhead of 111.7%. As the number of data communication phases increases, due to the same reason of the CMs in SeBCDCP-MAC and SeBCDCP-ENC having the least extra message overheads in the data communication phase, the CH's overhead also drops to 107.4% for SeBCDCP-MAC and 111.1% for SeBCDCP-ENC, and eventually will infinitely close to 101.6% and 102.4%. The relative message overhead of CH at the end of a CH-to-CH routing path is depicted in Figure 8.2 (b). For a single communication phase per setup phase, a SeBCDCP-MAC CH has an overhead of 104.0% and a SeBCDCP-ENC CH has an overhead of 106.0%. As the number of communication phases increases, the message overhead decreases slightly and it levels out at 103.8% for SeBCDCP-MAC and 105.7% for SeBCDCP-ENC.

Overall, it can be seen that increasing the number of communication phases per setup phase significantly decreases the relative overhead for CMs, while it only slightly decreases for CHs. As expected, the additional security of SeBCDCP-ENC has the price of a higher relative message overhead.

➤ *Varied Number of Clusters Performance Comparison*

The analysis assumes 50 communication phases per round ($N_{\text{DataCom}}=50$) and 50 members per cluster ($N_{\text{Member}}=50$). Obviously, the total number of clusters will only affect CH that forward data from previous CH on the CH-to-CH routing path. Figure 8.3 shows the relative message overhead of a CH at the end of the routing path: For a single cluster, a SeBCDCP-MAC CH has a message overhead of 106.5% and a SeBCDCP-ENC CH has an overhead of 109.8%. As the number of clusters increases, the relative overhead decreases and it levels out at 101.9% for SeBCDCP-MAC and 102.9% for SeBCDCP-ENC. Overall the analysis demonstrates that the relative performance of SeBCDCP-MAC and SeBCDCP-ENC improves with an increasing number of clusters. Again, the additional security of SeBCDCP-ENC has the price of a higher message overhead.

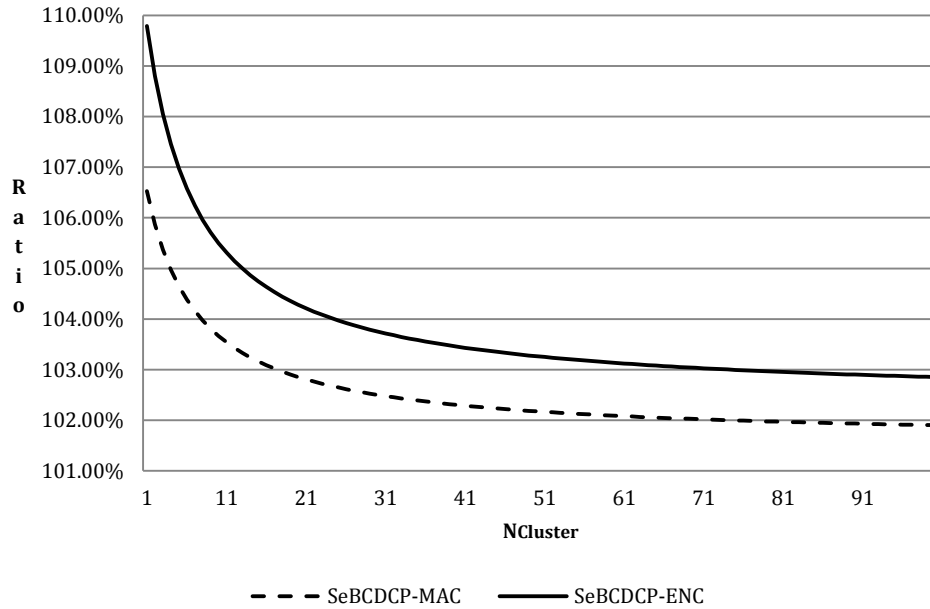


Figure 8.3 Relative Message Overhead of Last CH in CH-CH Routing Path, in dependency of $N_{Cluster}$

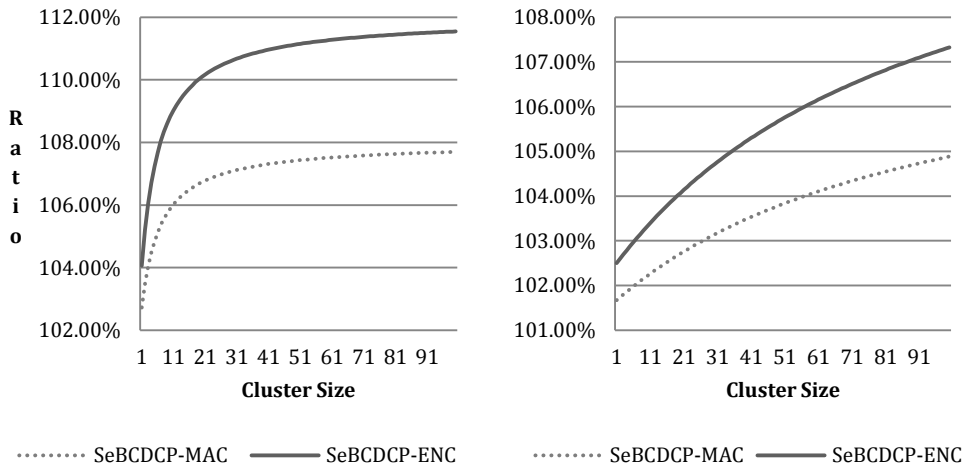
➤ Varied Cluster Size Performance Comparison

This sub-section compares the relative performance in dependency to the cluster size, i.e. the number of member nodes in each cluster. As the performance of CMs is not affected by the cluster size, the performance comparison for CMs is omitted. Further, 50 communication phases per round ($N_{DataCom}=50$) and a total number of 10 clusters ($N_{Cluster}=10$) are assumed.

Figure 8.4(a) displays the relative performance of a first CH on the routing path. For a single node cluster, a SeBCDCP-MAC CH has an overhead of 102.7%, whereas a SeBCDCP-ENC CH has an overhead of 104.1%. As the cluster size increases, the overhead also increases and it levels out at 108.0% for SeBCDCP-MAC and 112.0% for SeBCDCP-ENC. The last CH on a routing path needs to process data from previous CHs. As shown in Figure 8.4(b), for a single node cluster, the overhead is 101.7% for SeBCDCP-MAC and 102.5% for SeBCDCP-

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

ENC. Again, as the cluster size increases, the overhead also increases. For very large clusters of several thousand members it levels out at 108.0% for SeBCDCP-MAC and 112.0% for SeBCDCP-ENC.



(a). CH at the End of Routing Path

(b). First CH in Routing Path

Figure 8.4 Ratio of the Protocols Message Overhead, in Relation with the Cluster Size

Overall, the size of a cluster has a negative effect on the relative performance of SeBCDCP-MAC and SeBCDCP-ENC. Again, the additional security offered by SeBCDCP-ENC is paid for with a higher overhead.

As the figure shows, the increasing cluster size has negative energy consumption effects to the CHs running SeBCDCP-MAC and SeBCDCP. However, too large or too small cluster size may also shorten the network lifetime. In [Wan08], the author announced an equation to compute the optimal cluster size.

$$\text{Optimum cluster size } x = \sqrt[3]{2 * N}$$

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

Where N is the total of number of nodes. Thus, according to our assumption, $N = 500$, the optimum cluster size is 13.

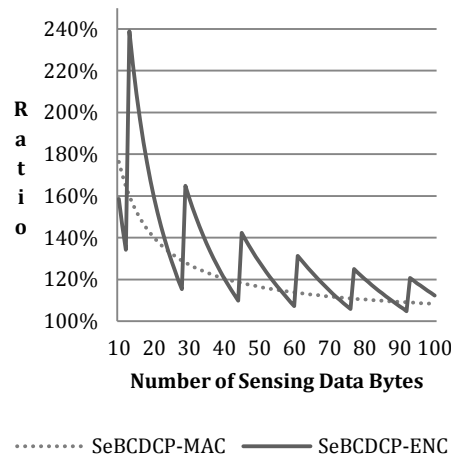
➤ *Varied Sensed Data Performance Comparison*

This sub-section compares the relative performance in dependency of the size of the sensed data. A cluster size of 50 ($N_{\text{Member}}=50$), 10 clusters ($N_{\text{Cluster}}=10$) and 50 data communication phases per round ($N_{\text{DataCom}}=50$) are assumed. Further, the analysis assumes that the data fusion process will compress data to a ratio of 1:5.

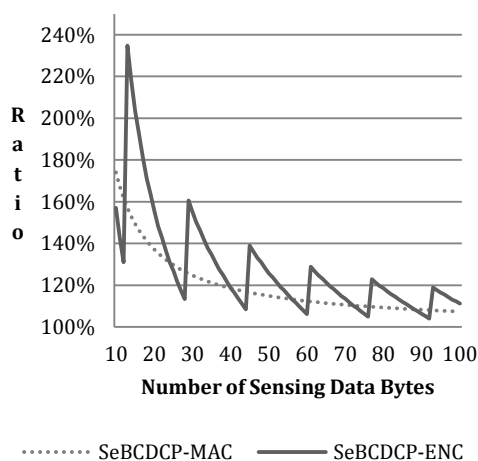
Figure 8.5(a) shows the relative performance of a CM. For a SeBCDCP-MAC CM the overhead is 176.4% for 10 bytes of sensing data. As the size of sensing data increases, the overhead decreases and it levels out at 108.3%. The spark line shape of SeBCDCP-ENC is caused by the byte padding characteristic of the block ciphers [Tuc97]: when the sensed data size rises to 12 bytes, the added nonce size of 4 bytes will make the total 4th message having the size of 16 bytes, equal to the AES block size, which makes the message has the extra message overhead of only 4 bytes. While when the sensed data size goes to 13 bytes, the encrypted message will have to be prolonged to be 32 bytes and causes an extra 19 bytes extra message overhead than BCDCP. The extra message overhead ratio will slightly drop along with sensed data length increases, and suddenly rise again when the sensed data size plus the nonce size equals to N ($N = 2, 3, 4, \dots$) times of the block size plus 1, and iterate. This also causes SeBCDCP-ENC having fewer messages overhead ratio at some points than SeBCDCP-MAC, as SeBCDCP-MAC has a constant 8-byte extra message overhead. Thus, the overhead of SeBCDCP-ENC is 238.8% at the worst case (13 bytes sensing data), but the average overhead decreases as the size of sensing data increases and it levels out at about 112.3%.

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

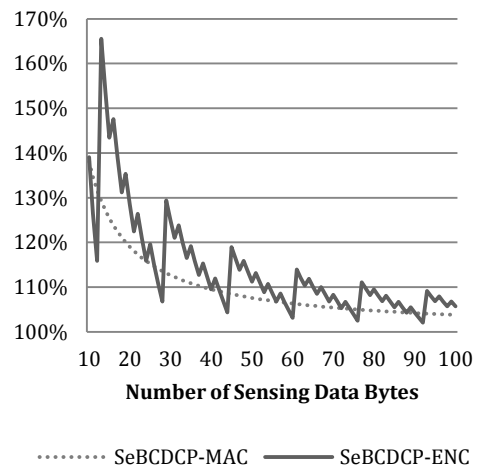
Figure 8.5(b) shows the relative performance of a first CH on the routing path. For a SeBCDCP-MAC CH the overhead is 174.1% for 10 bytes of sensing data. As the size of sensing data increases, the overhead decreases and it levels out at 107.4%. For SeBCDCP-ENC the relative is 234.8% in the worst case and levels out at 111.1%. Again, the spark line shape of SeBCDCP-ENC is caused by the encryption block size.



(a). CM



(b). CH at the End of Routing Path



(c). First CH in Routing Path

Figure 8.5 Ratio of the Protocols' Message Overhead, in Relation with the Sensing Data Size

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

A CH at the end of the routing path needs to forward data from previous CHs. Figure 8.5(c) shows the relative performance of Ch at the end of a routing path. A SeBCDCP-MAC CH has an overhead of 138.1% for 10 bytes of sensing data. As the size of sensing data increases, the overhead decreases and it levels out at 103.8%. In the worst case the overhead is 165.5% and the overhead levels out at 105.7% for large sensing data. The relative overhead of a SeBCDCP-ENC CH follows a ragged spark line shape, again caused by the block cipher: in the worst case, the CH to be evaluated has to receive all CMs' sensed data, and forward all other CHs' sensed data. Both the CMs and the CHs' extra message overhead changes will be reflected to the evaluated CH's ratio change.

Overall, the larger the size of sensing data, the better the relative performance of SeBCDCP-MAC and SeBCDCP-ENC. Even though in some cases the overhead of SeBCDCP-ENC is slightly less than the overhead of SeBCDCP-MAC (which is when the extra bytes the cipher has to transmitted are less than the size of MAC digest), in general the higher security of SeBCDCP-ENC has a price of a higher message overhead.

In general, the added security of SeBCDCP-MAC and SeBCDCP-ENC causes a message overhead compared to the original BCDCP protocol. Assuming a WSN which:

- Organises into 10 clusters with 50 members each
- Uses 50 data communication phases per round
- Has 100 bytes of sensing data per communication phase

Therefore the following extra message overhead is required:

- 8.32% for a SeBCDCP-MAC CM
- 12.32% for SeBCDCP-ENC CM
- A maximum 7.42% for a SeBCDCP-MAC CH

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

- A maximum 11.32% for a SeBCDCP-ENC CH

Considering the sensed data in 4th message has 100 bytes, CMs have 8% and 12% extra message overhead in SeBCDCP-MAC and SeBCDCP-ENC, respectively. The 50 data communication phases make their overall extra message overheads close to these values. For CHs, although their transmitted 5th step messages have the extra message overheads of only 1.6% for SeBCDCP-MAC and 2.4% for SeBCDCP-ENC, they also have to receive 50 members' 4th step message, which makes their overall extra message overheads close to 8% and 12%, respectively.

In general, both the SeBCDCP-MAC and SeBCDCP-ENC's extra message overhead will be reduced along with:

- The number of data communication phases per round increases
- The number of clusters in the network increases
- The size of the cluster decreases
- The size of the sensed data message, along with the size of the fusion data message increases.

SeBCDCP-MAC has less overhead than SeBCDCP-ENC. Although SeBCDCP-ENC has only around 4% more message overhead than SeBCDCP-MAC, it still means longer battery life. We assume WSN nodes are equipped with the battery with 3.7V and 1600mA power supply (2 AA batteries) and each round is iterated once a day. According to the computation results listed in [MMB05], the battery can keep the WSN alive for more than 769 rounds. Also according to my computed results showing above, WSN running SeBCDCP-MAC would have over 25 rounds longer network lifetime than SeBCDCP-ENC. A single MICAZ node has a retail price of \$150. Thus, in theory the WSN adopted SeBCDCP-MAC could save over \$2400 than the one adopted SeBCDCP-ENC. However, SeBCDCP-MAC only provides authentication, freshness and integrity protection.

8. Theoretical Estimation of WSN Routing Protocols Using Heinzelman's Model

At the cost of higher message overhead, SeBCDCP-ENC additionally provides confidentiality of sensing data. If the WSN application needs its network to be protected but the confidentiality of the sensed data is not that important, SeBCDCP-MAC surely is a better option than SeBCDCP-ENC.

9. Performance Analysis of Different AES Implementations

AES can be implemented on WSN nodes either exclusively in software or with hardware support. For the software implementations of AES, two algorithms are involved: the original Rijndael (AES) algorithm [DaR99] and the optimised algorithm that performs several operations as table lookups [BBF03]. Hardware support for AES is available on many platforms. For example the widely used RF transceiver module, Chipcon CC2420, provides a 128-bit AES encryption functionality [Chi04]. While an AES implementation with hardware support is faster than software implementations [HNL07]. It is not clear whether a faster implementation necessarily implies a more energy-efficient implementation, as the extra hardware requires additional power.

In general, this chapter presents a performance and energy consumption analysis of three AES implementations:

- Exclusive software AES using the original Rijndael algorithm.
- Exclusive software AES using an optimized table lookup AES.
- Hardware-supported AES using the Chipcon CC2420 RF transceiver chip.

The analysis examines the memory requirements, execution times and energy efficiency of the employed algorithms when implemented on a MICAZ mote [Cro12a] running TinyOS 2.1.0 [LMP05]. The power consumption is measured using the Agilent 66321D Mobile Communications DC source and 14565B Device Characterisation Software [Agi07].

9.1 Employed Systems

The presented analysis is performed using the widely-used MICAZ mote [GRL08] [KOK09] [XWD09] [BSP10] as an implementation and test platform (Cf. Figure 9.1). The MICAZ mote is a 2.4 GHz, IEEE/ZigBee 802.15.4 board used for low-power wireless sensor networks, with an 8-bit ATmega128L microcontroller, 128 KB RAM, 512 KB ROM and a 2.4 GHz Chipcon CC2420 RF transceiver. The Chipcon CC2420 features hardware AES-128 encryption support with multiple modes: standalone mode, counter mode (CTR), CBC-MAC authentication mode, and CCM encryption and authentication mode.

Power consumption is monitored using an Agilent 6321D Mobile Communications DC Source and the Agilent 14565B Device Characterisation Software. The Agilent 66321D can be used as a DC source for mobile devices such as MICAZ motes or cell phones. It provides an output voltage of 0–15 V and current of 0–3 A. While supplying energy to a device, it is able to sample output voltage and current with a time resolution ranging from 15 μ s to 31,200 s. The Agilent 14565B software has multiple functions:

- Controlling the Agilent 66321D to supply power with defined current or voltage.

9. Performance Analysis of Different AES Implementations

- Measuring and recording the current and voltage changes at the designated time resolution.
- Generating the current/voltage graph over time.

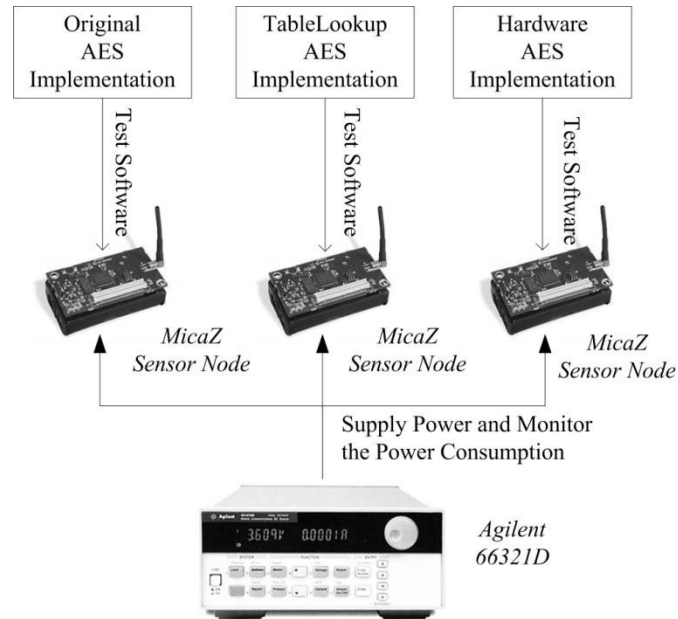


Figure 9.1 Analysis Outline

Combining the Agilent 66321D DC Source with the 14565B Device Characterisation Software, the mobile device's energy consumption can be monitored and recorded with high precision (resolution of 2.5 μ A for current measurements and 3 mV for voltage measurement). The MICAZ nodes run on TinyOS, which is the first operating system designed specifically for WSN [LMP05]. It is an open-source OS that is energy efficient and provides good power management features. For example, it automatically puts the processor into soft-sleep mode (using about 3.5 mA for the ATmega128L) when no task is waiting in the queue. For the analysis presented in this paper, TinyOS 2.1.0 is used. The programming language used to develop applications for TinyOS is nesC [GLB03], which is an extension of the C language. Furthermore, TinyOS

supports a wide range of different sensor platforms, offering interfaces to many of their specific features.

9.2 Implementation of AES Algorithms

The three AES algorithms *OriginalAES*, *TableLookupAES* and *HardwareAES* are implemented for TinyOS as user defined components. A unified interface is used for each of the variants as shown in Figure 9.2. Due to MICAZ's CC2420 radio transceiver lacks AES decryption functionality, the *HardwareAES* interface has no decryption command interface.

```
interface OriginalAES {
  command void AES_set_encrypt_key(const uint8_t *key);
  command void AES_encrypt(uint8_t *in, uint8_t *out);
  command void AES_decrypt(uint8_t *in, uint8_t *out);
}

interface TableLookupAES {
  command void AES_set_encrypt_key(const uint8_t *key);
  command void AES_encrypt(uint8_t *in, uint8_t *out);
  command void AES_decrypt(uint8_t *in, uint8_t *out);
}

interface HardwareAES {
  command void AES_set_encrypt_key(const uint8_t *key);
  command void AES_encrypt(uint8_t *in, uint8_t *out);
}
```

Figure 9.2 Interfaces of AES Implementations

The *OriginalAES* implementation is based on Niyaz's AES Implementation in C/C++ [Niy09], which follows the original AES algorithm. Some modifications have been incorporated to adapt it to the WSN node environment, such as reducing memory moves, using global variables and restricting the implementation to support 128-bit keys only. In addition, the source code is also modified to comply with the NesC format, containing components, modules, and interfaces. The whole procedure is divided into a key-setup process, an encryption process and a decryption process. The command *AES_set_encrypt_key* is used to derive the used round keys from the 128-bit key. The commands *AES_encrypt* and *AES_decrypt* can only be called after *AES_set_encrypt_key*, as otherwise the round keys are not correctly initialised. These commands encrypt/decrypt the content at the parameter *in*, and store the result at the parameter *out*.

As discussed in chapter 5.2.2, AES can be optimised by replacing some operations with table lookups. The *TableLookupAES* implementation is modified to comply with NesC format and the ATmega128L hardware. Furthermore, optimisations based on the recommendations published by Bertoni et al. [BBF03], Zhang and Niu [ZhN08] and Diala et al. [DAB08] are incorporated. The same interface and functions in the *OriginalAES* are used.

The *HardwareAES* implementation uses the CC2420's standalone AES encryption mode, as it is the only mode that does not involve radio communication or other extra features such as authentication. Analogous to the software implementations, the command *AES_set_encrypt_key* initialises the round keys. Since the key expansion computation is done by hardware, the *Hardware_AES* implementation's key-setup process involves setting security control registers to proper values and writing the 128-bit encryption key to KEY RAM in the CC2420

chip. The command *AES_encrypt* copies the content at the parameter 'in' to the SABUF RAM (specially designed RAM for storing a single plaintext block for encryption purposes) and starts the encryption. When the encryption process is finished, the content in SABUF RAM is replaced with the encrypted data. The *AES_encrypt* copies the encrypted data to the parameter *out*.

All three implementations were verified for correctness by matching their encryption/decryption results against the samples presented in FIPS-197.

9.3 Performance and Power Consumption Measurement

Test applications have been developed to analyse execution times, memory usage and power consumption of the key setup phase and the encryption/decryption phases of the AES implementations. To increase the accuracy of the measurements, these applications repeat each key setup, encryption, and decryption operation 100 times (no decryption operation for HardwareAES). The resulting average values are then scaled down to a single operation value. Measurements of the ROM and RAM usages are provided by TinyOS, which provides a function to display memory usage after successful compilation. The execution time and power consumption are measured with the Agilent 66321D and 14565B. The 66321D supplies the power to the MICAZ node and records the voltage and current supply/usage. The 14565B processes these records and provides corresponding graphs.

9.4 Results Analysis

This section discusses the results obtained from the performance and power consumption analysis performed, which was outlined in the last section. For all comparative figures, the values of *OriginalAES* are taken as the baseline.

9.4.1 Memory usage

The memory usage for the key-setup procedure of the three implementations is summarized in Table 9.1. There are only minor differences in ROM usage between the *OriginalAES* and *TableLookupAES* implementations, while *HardwareAES* requires more than twice the amount of ROM. On the other hand, *HardwareAES* uses the least amount of RAM (about 61% of *OriginalAES*). This is as expected, as most operations are provided by the hardware support of the CC2420 chip. For the software implementations, *TableLookupAES* requires only about 77% of the RAM of *OriginalAES*.

Table 9.1 ROM and RAM usage for Key Setup

<i>Implementations</i>	<i>ROM(byte)</i>	<i>RAM(byte)</i>
OriginalAES	7138 (100%)	1322 (100%)
TableLookupAES	6992 (98%)	1014 (77%)
HardwareAES	15282 (214%)	801 (61%)

Memory usage for encryption is summarized in Table 9.2, which is similar to key setup: *HardwareAES* uses the largest amount of ROM (156%), while *TableLookupAES* uses the least (87%); and *HardwareAES* uses the least amount of RAM (88%), while *TableLookupAES* requires the most (165%). Note that part of the higher ROM requirements for *HardwareAES* is due to the code to activate the

9. Performance Analysis of Different AES Implementations

CC2420 chip. As practical WSN applications need to activate the chip also for communication purposes, the ROM usage of applications, using any of the software AES implementations, would have a ROM requirement comparable to the hardware supported implementation.

Table 9.2 ROM and RAM usage for Encryption

<i>Implementations</i>	<i>ROM(byte)</i>	<i>RAM(byte)</i>
OriginalAES	9890 (100%)	946 (100%)
TableLookupAES	8642 (97%)	1558 (165%)
HardwareAES	15373 (156%)	833 (88%)

Table 9.3 depicts the memory usage for decryption. It can be seen that there are only minimal differences in the usage of memory between the two software implementations.

Table 9.3 ROM and RAM usage for Decryption

<i>Implementations</i>	<i>ROM(byte)</i>	<i>RAM(byte)</i>
OriginalAES	9320 (100%)	2392 (100%)
TableLookupAES	9496 (102%)	2326 (97%)
HardwareAES	n/a	n/a

9.4.2 Execution Time

A summary of the measured execution time for the key setup procedure of the three AES implementations is presented in Table 9.4. For the two software implementations, the key-setup procedure includes the expansion of the key into the keys required for each round, whereas for *HardwareAES* the procedure only includes copying the key to the CC2420's memory and the preparation of several

9. Performance Analysis of Different AES Implementations

registers on the chip. The expansion of the key is performed during the encryption process itself.

Table 9.4 Execution Times for Key Setup

<i>Implementations</i>	<i>Time (μs)</i>	<i>Ratio</i>
OriginalAES	317.4	100%
TableLookupAES	441.4	139.1%
HardwareAES	224.5	70.7%

It can be seen, that *HardwareAES* uses the least amount of time (70.7%). Among the software implementations, *OriginalAES* (100%) outperforms *TableLookupAES* (139.1%). The additional time in *TableLookupAES* is spent on preparing the employed tables.

Table 9.5 summarizes the measured execution times for the three AES implementations. As expected, *HardwareAES* outperforms the software implementations.

Table 9.5 Execution Times for Encryption

<i>Implementations</i>	<i>Time (μs)</i>	<i>Ratio</i>
OriginalAES	1104.3	100%
TableLookupAES	885.8	80.2%
HardwareAES (a)	33.2	3.0%
HardwareAES (b)	350.6	31.7%

If only the time required for encryption is considered, then *HardwareAES* requires only 3% of the time of *OriginalAES* (entry *HardwareAES* (a) in Table 9.5). This result is in line with previously published work [HNL07] [DAB08]. However, to encrypt data with hardware support, the plaintext must be copied to the CC2420 chip and the resulting cipher text must be copied back. The

software implementations do not require moving the data, as the encryption process can directly access the plaintext in memory. These measurements reveal that the copying of data between the MICAZ's RAM and the CC2420 SABUF RAM takes a significant amount of time. The total execution time of *HardwareAES*, when including the data transfer, rises to 350.6 μs (cf. entry *HardwareAES* (b) in Table 9.5): 151.8 μs are required to copy the plaintext from RAM to the CC2420, 33.2 μs are spent on encryption and 164.9 μs are used to copy the resulting cipher text back from the CC2420 to RAM. As a result, *HardwareAES* is only about three times faster than *OriginalAES*, and only about 2.5 times faster than *TableLookupAES*.

9.4.3 Energy Consumption

The Agilent 6321D Mobile Communications DC Source and the Agilent 14565B Device Characterisation Software are used to monitor the power consumption of the three AES implementations. For the key-setup procedure, the measurements are summarized in Table 9.6, which depicts the current measurement taken, where a voltage of 3 V is used. Measurements are taken for 100 operations, which are then scaled down to the single-operation values shown. It can be seen, that while *HardwareAES* performs faster than the software implementations, it requires more than 3.5 times the current.

Table 9.6 Energy Consumption of Key Setup

<i>Implementations</i>	<i>Time (ms)</i>	<i>Current (mA)</i>	<i>Energy (μJ)</i>	<i>Ratio</i>
OriginalAES	0.3174	7.075	6.74	100%
TableLookupAES	0.4414	7.162	9.48	141%
HardwareAES	0.2245	25.414	17.12	254%

9. Performance Analysis of Different AES Implementations

Overall, the key-setup procedure of *HardwareAES* consumes more than 2.5 times the energy of *OriginalAES* and nearly double the energy of *TableLookupAES*. The additional energy consumption of *HardwareAES* is due to the power required to power the CC2420 chip.

Similar results are obtained, as shown in Table 9.7 and Table 9.8, for the energy consumption of encryption and decryption respectively. While *HardwareAES* is the fastest implementation in encryption single block data, it uses more energy than either software implementation (14% more than *OriginalAES* and about 40% more than *TableLookupAES*). However, in terms of encryption *TableLookupAES* is the most energy-efficient implementation, as it uses nearly 20% less energy than *OriginalAES*. On the decryption side, *TableLookupAES* performs even better as it requires only 41% energy of *OriginalAES* to decrypt a block data. CC2420 chip cannot decrypt the message, as it is not included in the comparisons displayed in Table 9.8.

Table 9.7 Energy Consumption of Encryption

<i>Implementations</i>	<i>Time (ms)</i>	<i>Current (mA)</i>	<i>Energy (μJ)</i>	<i>Ratio</i>
OriginalAES	1.1043	7.116	23.57	100%
TableLookupAES	0.8858	7.211	19.16	81%
HardwareAES	0.3506	25.501	26.82	114%

Table 9.8 Energy Consumption of Decryption

<i>Implementations</i>	<i>Time (ms)</i>	<i>Current (mA)</i>	<i>Energy (μJ)</i>	<i>Ratio</i>
OriginalAES	2.8239	6.914	58.62	100%
TableLookupAES	1.1371	7.2098	24.58	41%
HardwareAES	n/a	n/a	n/a	n/a

9. Performance Analysis of Different AES Implementations

Figure 9.3 depicts the dependency of the relative energy consumption, in relation to the frequency of key changes. In the case where the key is changed for every encryption process, *HardwareAES* uses 45% more energy than *OriginalAES*, while *TableLookupAES* uses only 94.5% of *OriginalAES*.

As the number of encryptions per key setup increases, the relative energy efficiency of *HardwareAES* and *TableLookupAES* improves. In the limit, *HardwareAES* requires about 13.8% more energy than *OriginalAES*, while the energy consumption of *TableLookupAES* decreases to 81.3% of *OriginalAES*.

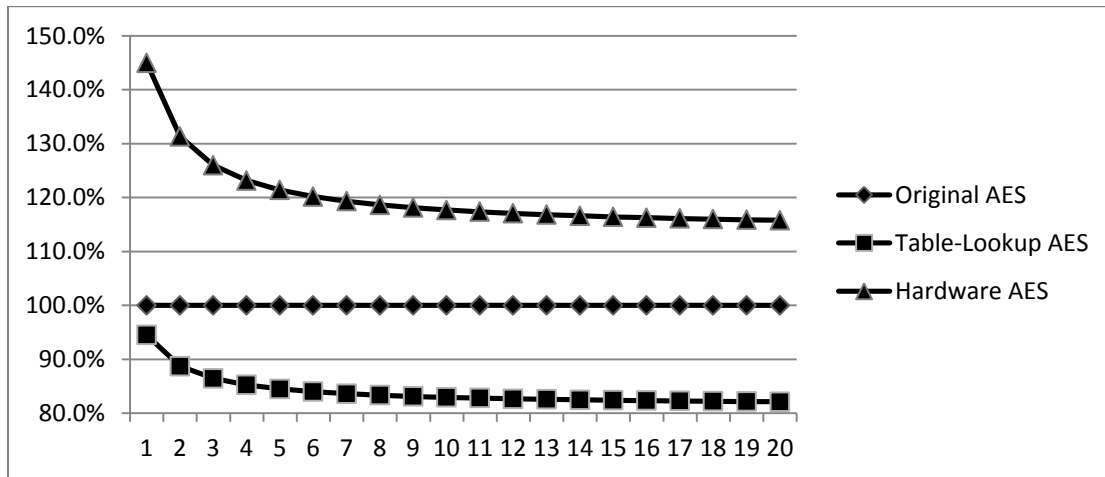


Figure 9.3 Overall Relative Energy Consumption

9.4.4 Which Implementation should be adopted?

From the point of view of time efficiency, *HardwareAES* implementation's key-setup procedure is about 30% faster than the fastest software implementation, and encryption is about 2.5 times faster than the best implementation. However, for most WSN applications the lifetime of the network relates deeply to the lifetime of the WSN node's power supply. Thus, energy consumption, rather than execution time, is often the decisive feature when a choice of multiple

solutions is available. The presented analysis has revealed that the speed advantage of hardware support is contrasted by its increased energy consumption. The key setup procedure of *HardwareAES* uses 2.5 times more than the *OriginalAES*, and nearly double the energy of *TableLookupAES*. Furthermore, the encryption process of *HardwareAES* uses 14% more energy than *OriginalAES* and about 40% more than *TableLookupAES*. If the entire process, consisting of key setup and encryption is considered, the results depend on the relative frequency of encryptions per key setup. Therefore, if the key is changed for every encryption process, *HardwareAES* is quite inefficient, as it uses 45% more energy than *OriginalAES* while *TableLookupAES* uses 94.5% of the *OriginalAES*. As the number of encryptions per key setup increases, the relative efficiency of *TableLookupAES* and *HardwareAES* increases to 81.3% and 113.8%, respectively. As Figure 9.3 shows, to minimize the energy consumption but not compromise the security, the key setup should be carried out after at least 16 encryptions / decryptions. Overall, while *HardwareAES* is the fastest implementation, it requires more energy than the software implementations. In conclusion, *TableLookupAES* is the most energy efficient AES implementation. Consequently, a node's lifetime can be significantly extended by choosing this implementation.

9.5 Comparison of Xiao's Model Estimation Results and the Empirical Measurement Results

Xiao et al. found a way to simulate the actual processor overhead while encrypting and decrypting a block of data using AES algorithm [XCS06]. In their research, they analyzed the algorithm of AES, and deduced the minimum number of processing cycles required for performing an AES encryption and decryption, shown as Eq. 9.1 and Eq. 9.2.

9. Performance Analysis of Different AES Implementations

$$T_E = (8BT_{and} + 4BT_{or}) + [46BT_{and} + (31B + 12)T_{or} + (64B + 96)T_{shift}](R - 1) + (8BT_{and} + 7BT_{or} + 3BT_{shift}) \quad (\text{Eq. 9.1})$$

$$T_D = (8BT_{and} + 4BT_{or}) + (8BT_{and} + 7BT_{or} + 3BT_{shift}) + [161BT_{and} + (122B + 12)T_{or} + (32B + 96)T_{shift}](R - 1) \quad (\text{Eq. 9.2})$$

Where T_E and T_D denote the minimum number of processing cycles for encrypting and decrypting block data, respectively; $4B$ is the block size (in bytes), R denotes the number of rounds, and T_{and} , T_{or} , and T_{shift} are the number of processing cycles for a processor to perform single byte-wise AND, byte-wise OR, and byte-wise SHIFT operations respectively.

Further, they assumed $T_{and} = T_{or} = T_{shift} = 1$ hold, $B = 4$ holds as the AES adopts the block size as 128 bits, and $R = 10$ for 128-bit key size. In conclusion, they calculated that the overheads for encrypting and decrypting a block for 128-bit key are 59us, and 123us for a 100MIPS processor.

Table 9.9 Computational and Real World Measurement Results Comparison

Implementations	Time Spending (μs)	Estimation Error (%)
<i>Measured Original AES Encryption</i>	1104.3	0.00%
<i>Estimated Original AES Encryption</i>	735.7	-33.38%
<i>Measured Original AES Decryption</i>	2823.9	0.00%
<i>Estimated Original AES Decryption</i>	1537.5	-45.55%

By utilizing Xiao's mathematical model, the durations that MICAZ mote spends on encrypting and decrypting single block data using AES algorithm are computed. However, the measurement results [ZDC11] in relation to the real world implementation, show relative big differences. Compared against the computational results, Table 9.9 displays the real world measurements of AES, which shows 50.1% more time is spent on encryption and 83.7% on decryption.

10. Implementation and Performance Analysis of Wireless Sensor Network Routing Protocols

Base-Station Controlled Dynamic Clustering Protocol (BCDCP) is a cluster-based energy efficient protocol. In BCDCP, the BS makes the clustering and routing sequence decisions, thus the computational burden of the WSN nodes is relieved. Two secured versions of BCDCP, namely SeBCDCP-MAC and SeBCDCP-ENC are introduced in Chapter 6. Both protocols keep the characteristics of BCDCP but add security protections.

In Chapter 8, the performance difference between BCDCP, SeBCDCP-MAC and SeBCDCP-ENC has been evaluated in theory, and concluded both secured protocols need no more than 15% extra message overheads than the original BCDCP, and that SeBCDCP-MAC has a better performance than SeBCDCP-ENC. This chapter details the actual implementation and measured performance analysis of the three protocols on a small experimental WSN, composed of a few MICAZ nodes: one act as BS, one as the CH, and the rest as its CMs.

As with the performance analysis of the implementation of AES (Cf. Chapter 9), the analysis is involved in the examination of the memory requirements, execution times and energy efficiency of the employed implementation.

10.1 Employed System

The presented implementation and analysis is composed of several MICAZ motes and one BS, shown as Figure 10.1. TinyOS requires each WSN node including the BS to be specified as an individual ID. The BS should have a static ID of 0X0000 [LMP05]. The WSN nodes can be specified from 0X0001 and thereafter.

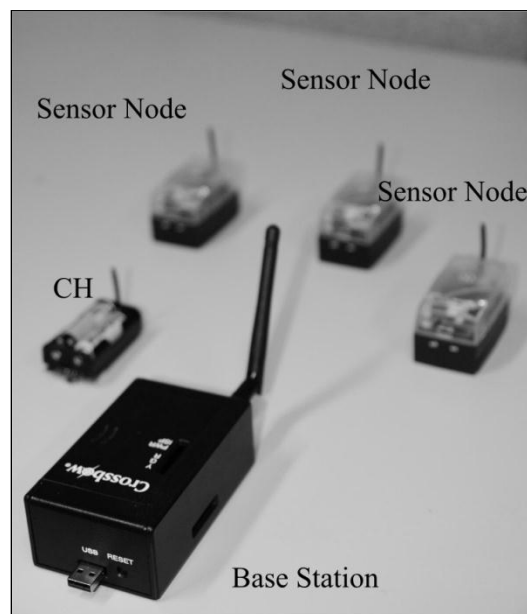


Figure 10.1 A WSN Composed of MICAZ motes and One Base Station

The proposed implementation of BCDCP can be applied to a small experimental WSN of one BS and up to 32 WSN nodes. The network deployment area can be divided into at most 256 areas (each with an individual Location ID). Each cluster can contain up to 6 WSN nodes plus a CH.

10.1.1 Differences between the Implemented Protocols and the Original Protocols

Due to the hardware limitation of MICAZ, the implementation design is slightly different than the original BCDCP, SeBCDCP-MAC, and SeBCDCP-ENC protocols. MICAZ cannot detect its remaining energy level in hardware, thus the only way of estimating the energy level is via software. It is assumed that all nodes are pre-installed with batteries at the same energy level, which is possible when each node is loaded with fully charged new energy sources, and with the same power volume. MICAZ's hardware specification [Cro12a] contains its current draw in different modes. It is possible to use a timer to record the time for the MICAZ staying on these modes, and then compute the consumed energy. Based on this assumption, a variable *consumed_energy_level* is maintained to log the overall energy consumption.

Secondly, the implementation of BCDCP does not contain the data sensing and data fusion processes. I assume the target WSNs adopting BCDCP, SeBCDCP-MAC, and SeBCDCP-ENC will collect same data in the same area, thus the energy consumed on data sensing and fusion should be the same. Neglecting the data sensing and fusion processes should not affect the energy consumption comparison results. However, the sensed data and fusion data should exist for transmission and receiving. Thus I pre-store a number of bytes in the WSN nodes as the simulated sensed and fusion data. They will be processed (being encrypted or used for generating MAC digest) and transmitted to the CH or the BS.

Lastly, to reduce the CMs' energy consumption in the listening mode, the 3rd message in BCDCP, SeBCDCP-MAC, and SeBCDCP-ENC is sent prior to the 2nd

message. This can be done, as the order in which these messages are sent does not impact on the operation of the protocol.

10.1.2 Packet Structure

The general data packet structure of the three protocols is defined as follows

```
typedef nx_struct {
    nx_uint8_t type;
    nx_uint8_t content[PRTOCOL_X_BUFFER_SIZE];
} ProtocolPacket_x;
```

In the *ProtocolPacket_x* structure, the *type* is on behalf of the message type of the next 16 bytes stored in *content*. There are 6 predefined types, representing the five steps shown in Figure 3.3, Figure 6.1, and Figure 6.2, and an extra *MAIN_TIMER_START_BEACON* message type, representing the message for activating or restarting the main timer. The constant *PRTOCOL_X_BUFFER_SIZE* is the number of bytes contained in the protocol message buffer.

10.1.3 Implementation Work Scheme

The objective of the implementations is to measure the energy consumption on the protocols' single setup phase and data communication phase. Thus the workflow of the implementation is simplified: the functionalities of the WSN nodes are implemented in full (except for the data sensing and data fusion), while the BS implementation is not: the BS will only assign the roles to two WSN nodes: one acts as CH and the other as its CM. Thus, the experiment network is grouped by three MICAZ motes: one act as BS, one as the CH, and one as the CM.

However, to simulate the cluster has 6 CMs and 1 CH, while measuring the energy consumption of the CHs, the CM iterates its 4th message send for 6 times, to simulate the circumstance of the cluster containing 6 members.

The simplified BS leads to a predigested workflow: upon startup, the BS first broadcast a start beacon to activate the timer of both WSN nodes. The two nodes then send the 1st message to the BS. BS will respond the message with 3rd message and then 2nd messages sequentially. The nodes parse the received message and then go to sleep. Here the setup phase ends.

After a short period, the CH and CM wake up at roughly the same time (thanks to the synchronized timer). The data communication phase now starts. CH turns on its radio transceiver and waits for the CM's 4th message. After the 4th message is received, the CH generates the 5th message and sends to the BS. Both nodes fall asleep again after their sending task is finished.

The designated workflow of the implementations is shown in Figure 10.2 and Figure 10.3. Each rectangle in the figures represents an action: encryption / decryption to a message, sending or receiving a message, etc. The rectangles with italic texts are the actions involved in security implementations, which is not executed in BCDCP implementation. The BS, CH, and CM's actions are placed in different columns, listed sequentially by time.

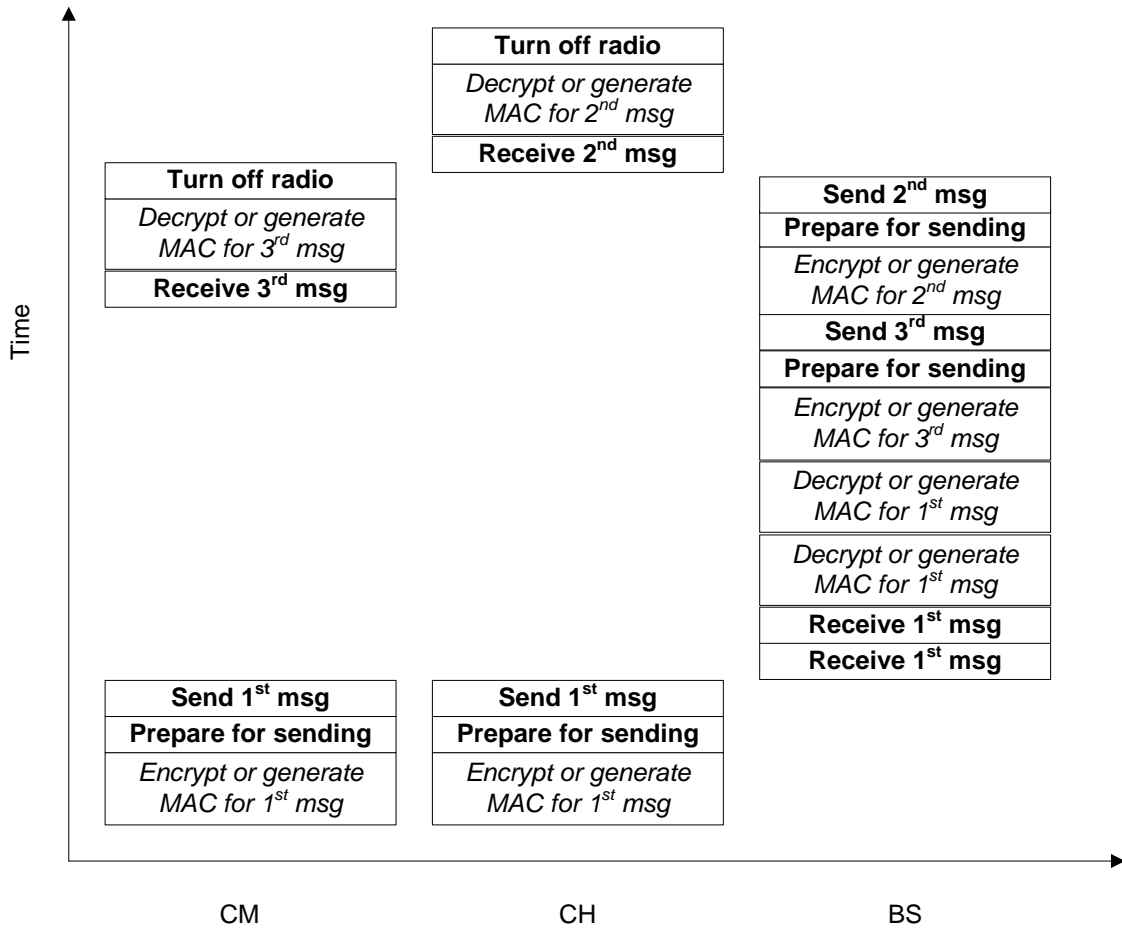


Figure 10.2 Workflow of the Protocols' Implementation in Setup Phase

Figure 10.3 involves only the CH and CM's actions: two WSN nodes turn on their radio transceiver simultaneously, and the CM then sends its sensed data to the CH. This procedure is iterated 6 times, to simulate the CH has 6 members in its cluster. After the CH has received 6 4th messages or 500ms is up, it sends the 5th message to the BS.

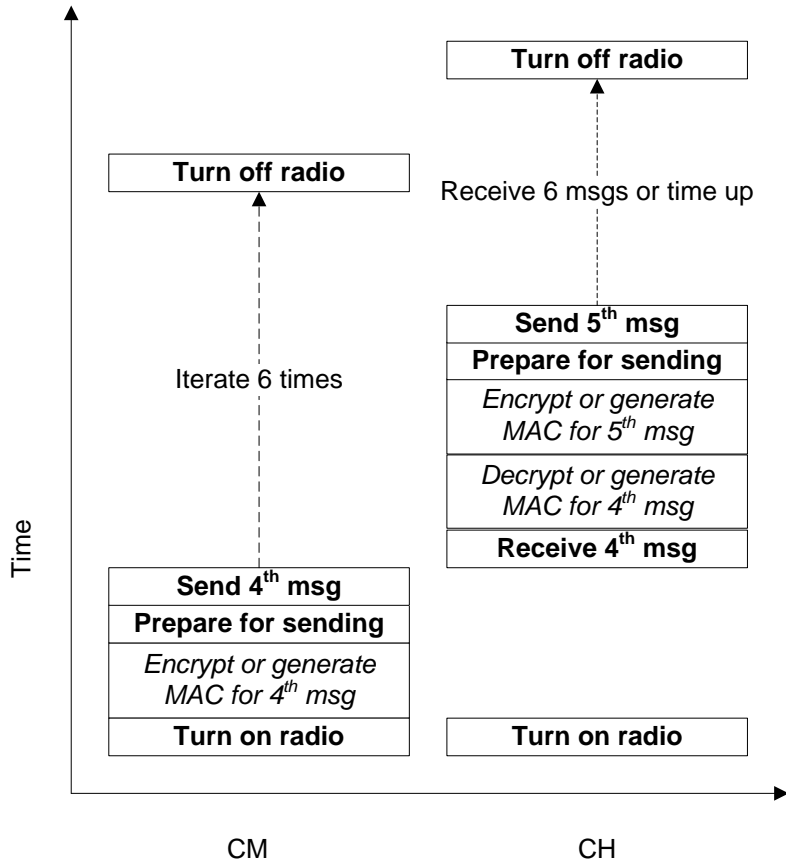


Figure 10.3 Workflow of the Protocols' Implementation in Data Communication Phase

10.2 Implementing BCDCP

The five steps' payload formats and the actual payload sizes of BCDCP implementation is presented in Figure 10.4. In the BCDCP implementation, the packet buffer sizes for each message are 33 bytes (1 byte type identifier and 32 bytes data). This may waste some energy on the radio transceiver side, but reduces the program complexity and increases the application stability: if too many lines of code is included in TinyOS's message receiving handler (called `Receive.receive()`), like applying different size memory for the packets with different lengths, I have found a larger probability of program crashes.

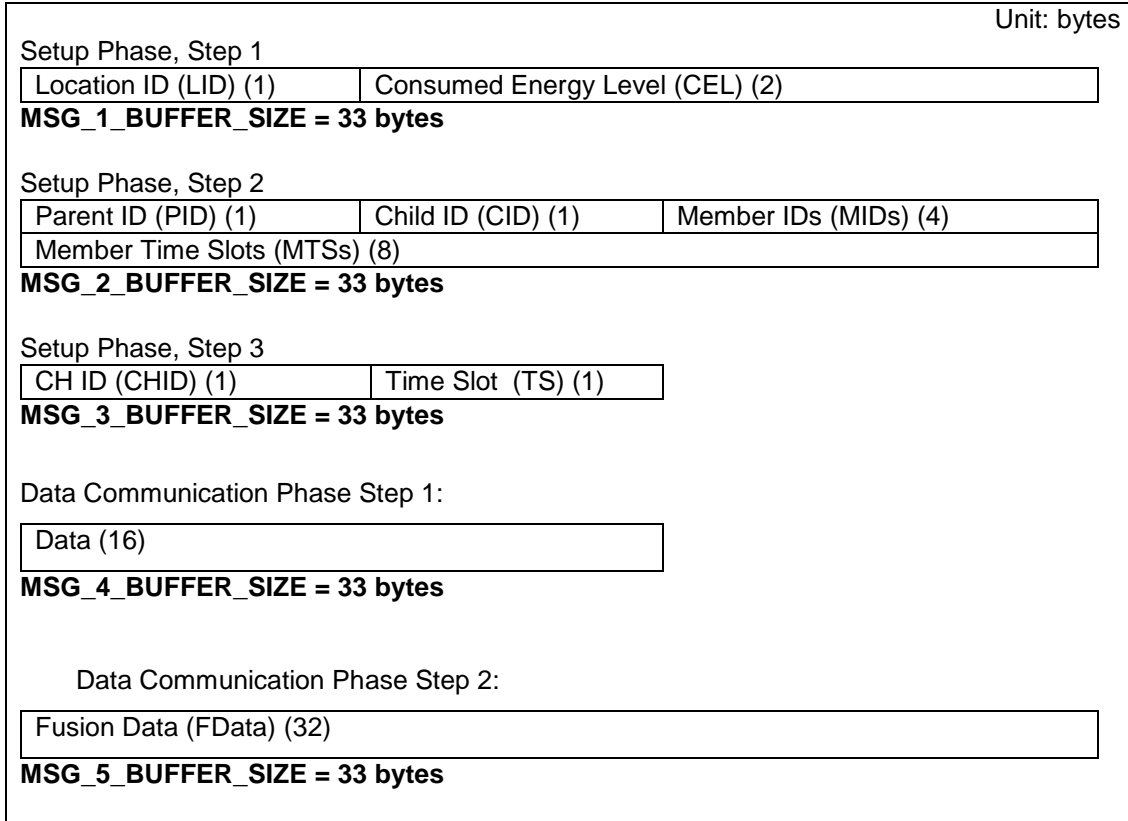


Figure 10.4 Packet Formats of BCDCP Implementation

In step 1, an 8-bit location ID (*LID*) can identify up to 256 location areas. In step 2, *PID* and *CID* is the receptor CH's parent and child ID in the CH-CH routing path. Once the receptor CH is assigned to locate at the end of the CH-CH routing path, it should receive a *CID* with value "0xFF", indicating that no child exists; while the first CH in the routing path receives a *PID* of "0x00" (BS's address in TinyOS). The Member ID set (*MID*) in step 2 uses a bit-oriented ID identification. For instance, assuming the BS decides that the nodes No.2, No.3, and No.4 are the node No.1's CMs, the *MID* that node No.1 receives would be "00000000,00000000,00000000,00001110" in binary, or "0x000E" in hexadecimal, which indicates the BS sets 2nd, 3rd, and 4th bits in *MID* as 1, and the rest as 0.

As I mentioned in the beginning of this chapter, the implementation does not contain the data fusion operation. However, I still assume the data fusion of the

CHs compress the received sensed “raw” data into a 32-byte message. This strategy also applies to SeBCDCP-MAC and SeBCDCP-ENC implementations.

10.2.1 Implementation’s Event Handler and Work Flow

Figure 10.5 depicts the *Receive.receive()* event handling in BCDCP implementation. This event is triggered when the radio chip receives a packet. The initial procedures of all nodes are set to keep the radio on until the first *Main_Timer_Start_Beacon* message is received. The Main Timer in the program is used to synchronize the packets’ transmission and reception in the data communication phase, and the RF transceiver’s on and off moment. The rest of the *Receive* event handling deals mainly with the steps’ message receptions. The corresponding functions are based on the value of type in received *BCDCPPacket*.

Figure 10.6 describes the Main Timer Trigger Event Handler. The Main Timer is used to synchronize the node to switch the RF transceiver on and off when the specific time for transmitting / receiving data message arrives. The Main Timer is set to fire every 10ms, which is the duration of one time slot. An integer *MainTimerFireTimes* is incremented by one when every Main Timer event is caught. CMs only need to switch on the RF transceiver when its time slot for transmitting the simulated sensed data comes. But the CHs have heavier duties for receiving the data packets that come from all their members, and transmitting the simulated fusion data to the BS. Both CH and CM stop the Main Timer when the data transmission and reception tasks are finished.

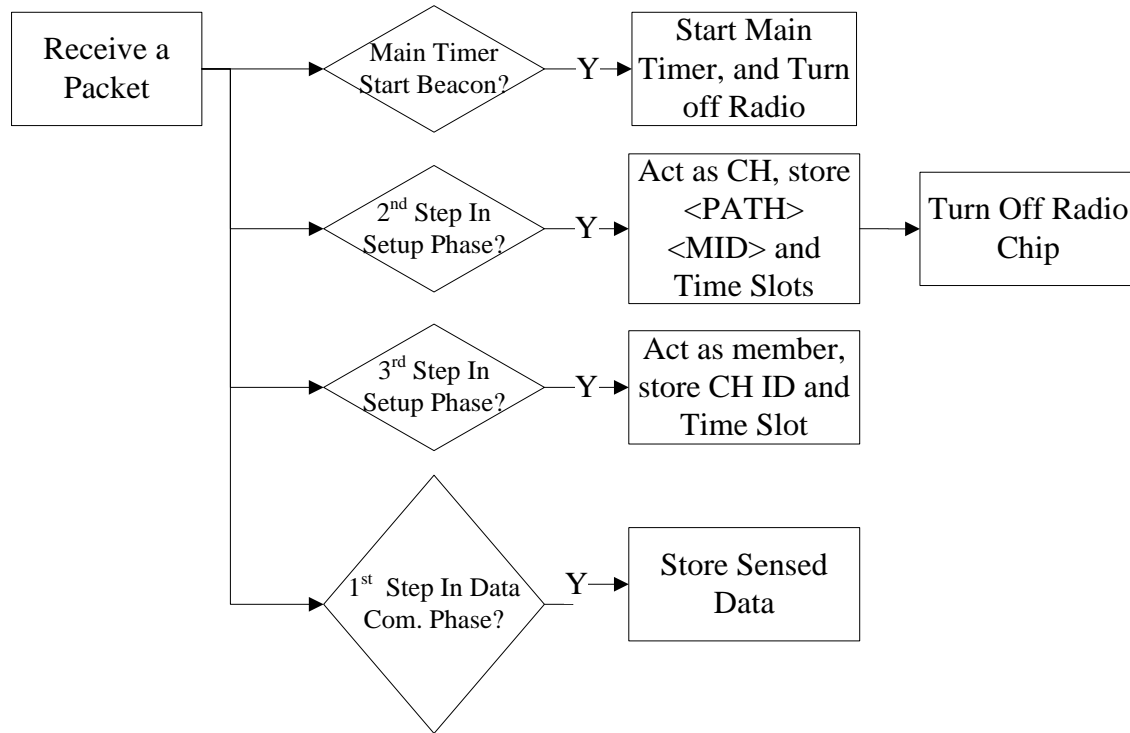


Figure 10.5 The Receive Event Handling Flow Chart

The Main timer trigger event handler does not deal with the detailed data packet transmission task. This is done by the *RadioControl.startDone()* event handler. This handler is activated when the RF transceiver is started. According to the current value of *MainTimerFireTimes*, the handler will decide which packet is to be transmitted, or does nothing but wait for the incoming packets, while the radio transceiver is kept on.

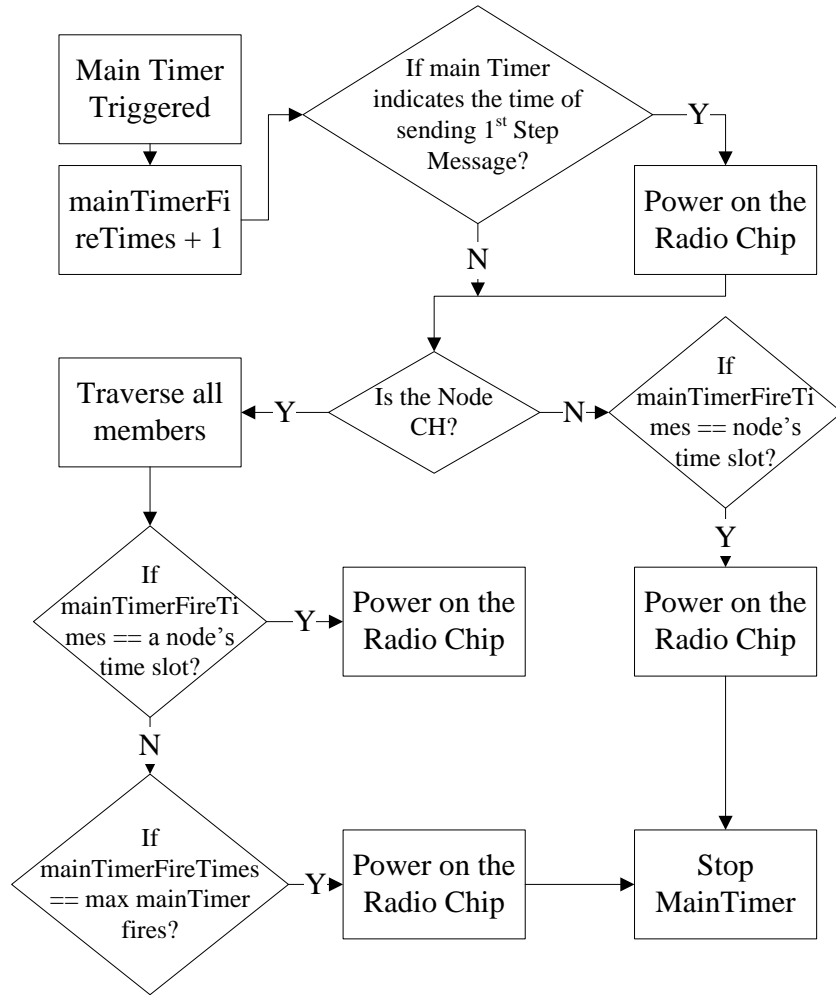


Figure 10.6 MainTimer Trigger Event Handling Flow Chart

When one sending or receiving task is finished, the program will power off the RF transceiver in order to save power. Directly calling *RadioControl.stop()* from the radio chip relative handler (i.e. the *send.sendDone()* event handler) in TinyOS would not result in the radio transceiver powering off, as it is still occupied by other threads. In our solution, a timer called *RadioOffTimer* is activated in the last step of the sending or receiving task. The timer is triggered in only one millisecond, in order to minimize any unnecessary power consumption. The RF transceiver's shutting down task is processed in the timer's event handler.

10.3 SeBCDCP-MAC and SeBCDCP-ENC Implementations

In the same way as with the BCDCP implementation, the SeBCDCP-MAC and SeBCDCP-ENC implementations do not contain the data sensing and data fusion implementations. Further, as all tests were executed in a controlled environment, accidental or malicious inference could not occur. Thus, the desynchronisation steps of the protocols (steps 2b, 3b) were not implemented.

The implementations of SeBCDCP-ENC and SeBCDCP-MAC require symmetric encryption and MAC generation computations. The proposed implementation adopts the *TableLookupAES* implementation as both the encryption and MAC generation (using AES-CBC-MAC) computations do, due to its proven energy efficiency and security [ZDC11].

In the same way as the analysis platform was introduced in Chapter 9, the implementations' power consumption is monitored using an Agilent 6321D Mobile Communications DC Source and the Agilent 14565B Device Characterisation Software.

In order to realize security protection in both implementations, the following necessary elements to the implementation are added:

Block Cipher and MAC digest generation. Table-Lookup AES implementation [ZDC11] is used in SeBCDCP-ENC and SeBCDCP-MAC: SeBCDCP-ENC uses it for encrypting and decrypting messages, and SeBCDCP-MAC employs it to encrypt messages and generate MAC digests. The implementation uses 128-bit keys and operates on a 128-bit data blocks. AES-CBC-MAC operates much more

10. Implementation & Performance Analysis of WSN Routing Protocols

efficient than other MAC algorithms like HMAC-MD5 [Dob96] and HMAC-SHA1 [EaJ01], as AES was designed to work efficiently even on weak processors like MICAZ's 8-bit ATmega128 microcontroller [DaR99], yet HMAC-MD5 and HMAC-SHA1 mainly operates on 32-bit words, which is inefficient on 8-bit processors: the 8-bit processor needs at least 4 times of CPU cycles than 32-bit processor, to read, write, or operate a 32-bit word.

Nonce. TinyOS's standard library, *RandomMlcgc* component is used to generate 16-bit nonce value, and the current consumed energy value as the seed.

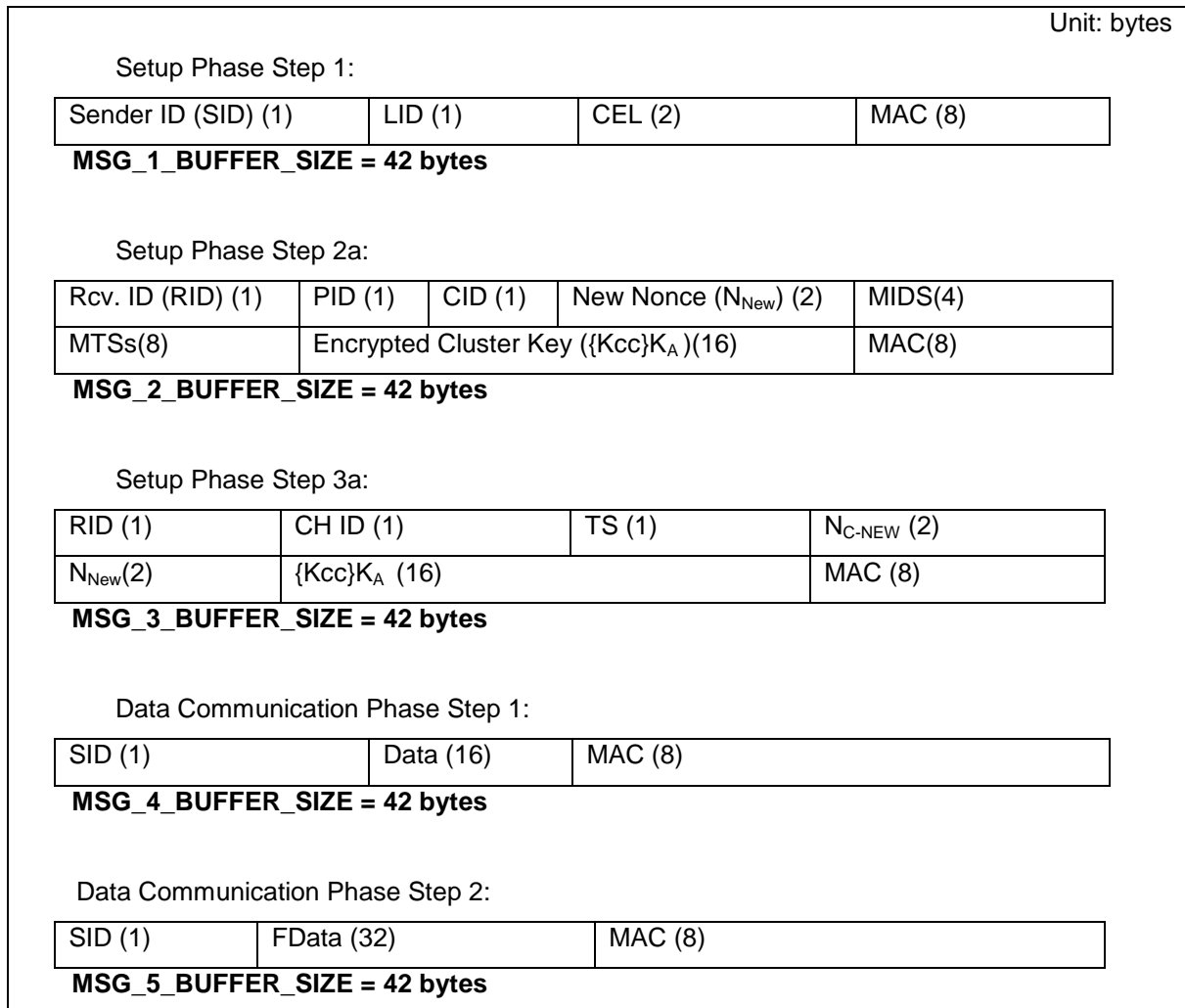


Figure 10.7 SeBCDCP-MAC Implementation Packet Format

10. Implementation & Performance Analysis of WSN Routing Protocols

The packet format and message sizes of SeBCDCP-MAC and SeBCDCP-ENC protocols are detailed in Figure 10.7 and Figure 10.8. For the same reason of preventing the MICAZ mote crashing, both implementations have payload buffers with three different sizes to hold the incoming and outgoing messages.

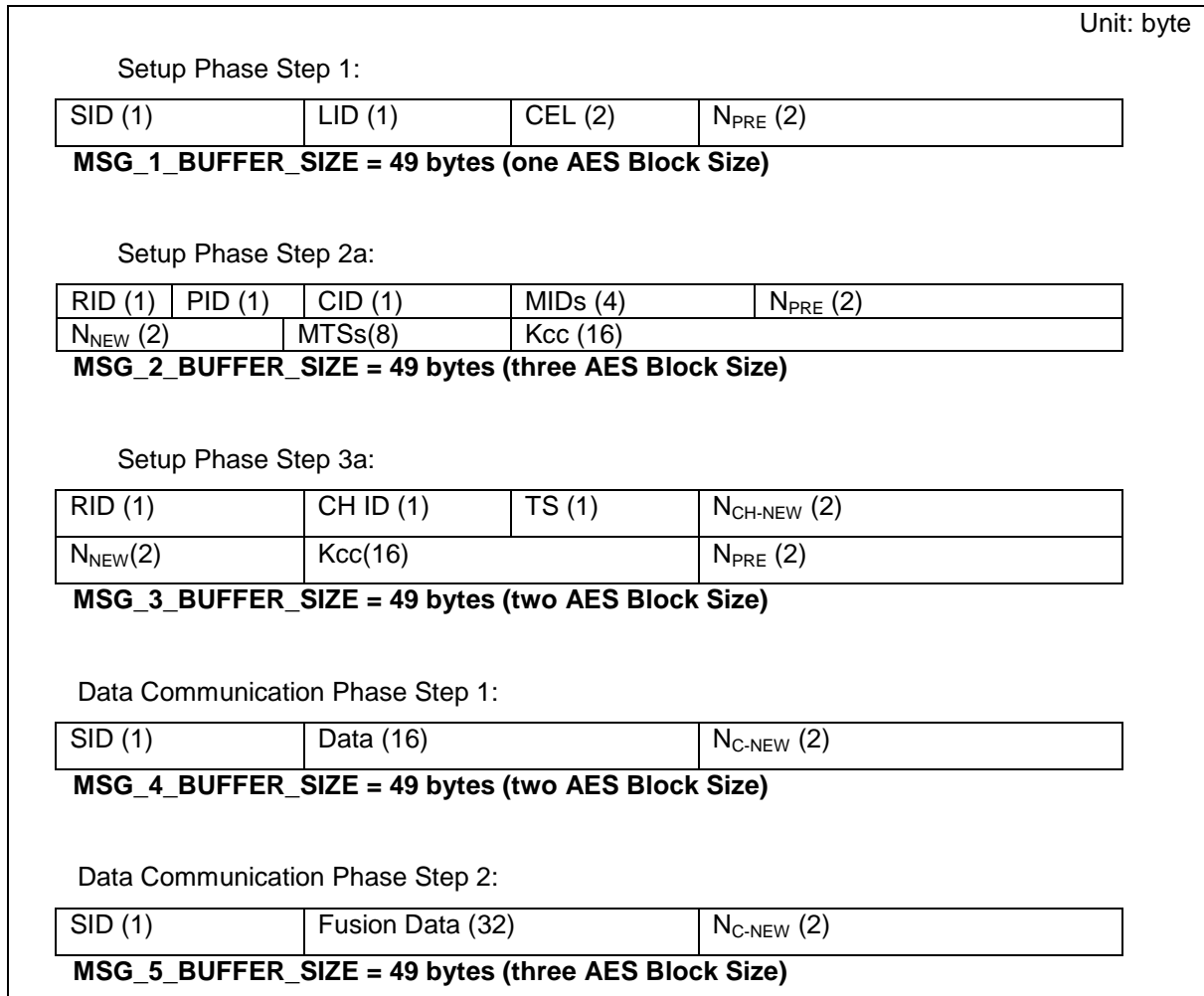


Figure 10.8 SeBCDCP-ENC Implementation Packet Format

SeBCDCP-MAC and SeBCDCP-ENC implementations have a similar structure to BCDCP implementation. The differences are in the packet formats, the extended message size, and the added security protections. The Table-Lookup AES implementation's source code introduced in Chapter 9 can be used for developing the two implementations, as it has been evaluated as most energy

efficient AES implementation among the three. The added programs include the encryption/decryption procedure (for SeBCDCP-ENC) and MAC digest generation (SeBCDCP-MAC, by using AES-CBC-MAC), plus some necessary security verification process (e.g. matching the receiver node ID and nonce, received and generated MAC digest, etc.).

BCDCP, SeBCDCP-MAC, and SeBCDCP-ENC's implementations have been proven to work correctly. Both the CHs and CMs respond to the incoming messages and complete the setup as well as the data communication phases well.

10.4 Empirical Measurement Performance Comparison among the Implementations

The Agilent 6321D Mobile Communications DC Source and the Agilent 14565B Device Characterisation Software are used to monitor the power consumption of the three routing protocols' implementations. Table 10.1 and Table 10.2 respectively show the CH's and CM's time and energy consumption measurement. In the tables, BCDCP's time and energy consumption show that the CH has to consume more time and energy to complete the setup phase; in SeBCDCP-ENC it has around 32% more time and energy consumption than BCDCP. This ratio rises to over 68% for SeBCDCP-MAC. For CMs, the cost of security drops to 14.12% of SeBCDCP-ENC and 34.13% of SeBCDCP-MAC.

In the data communication phase, as the experimental WSN contains one cluster with one CH and 6 CMs, in each round the CH receives 6 data packets from its members. SeBCDCP-ENC spends 32.67% more energy for security, and SeBCDCP-MAC, 33.15% (Cf. Table 10.3). For CMs, the cost of security in SeBCDCP-ENC is 48.42%, and 51.42% of SeBCDCP-MAC (Cf. Table 10.4).

10. Implementation & Performance Analysis of WSN Routing Protocols

Table 10.1 CH Setup Phase Time and Energy Measurement Result

<i>Implementation</i>	<i>Time (ms)</i>	<i>Ratio</i>	<i>Energy (mJ)</i>	<i>Ratio</i>
BCDCP	43.71	100%	2.69	100%
SeBCDCP-ENC	57.10	130.63%	3.55	131.72%
SeBCDCP-MAC	69.98	160.09%	4.55	168.84%

Table 10.2 CM Setup Phase Time and Energy Measurement Result

<i>Implementation</i>	<i>Time (ms)</i>	<i>Ratio</i>	<i>Energy (mJ)</i>	<i>Ratio</i>
BCDCP	33.00	100.00%	1.99	100.00%
SeBCDCP-ENC	36.66	111.09%	2.27	114.12%
SeBCDCP-MAC	42.10	127.58%	2.67	134.13%

Table 10.3 CH Data Communication Phase Time and Energy Measurement Result

<i>Implementation</i>	<i>Time (ms)</i>	<i>Ratio</i>	<i>Energy (mJ)</i>	<i>Ratio</i>
BCDCP	73.88	100.00%	4.70	100.00%
SeBCDCP-ENC	94.81	128.32%	6.24	132.67%
SeBCDCP-Mac	95.10	128.72%	6.26	133.15%

Table 10.4 CM Data Communication Phase Time and Energy

<i>Implementation</i>	<i>Time (ms)</i>	<i>Ratio</i>	<i>Energy (mJ)</i>	<i>Ratio</i>
BCDCP	10.27	100.00%	0.54	100.00%
SeBCDCP-ENC	13.94	135.74%	0.8	148.42%
SeBCDCP-Mac	14.35	139.72%	0.82	151.47%

In general, both SeBCDCP-MAC and SeBCDCP-ENC requires a certain amount of extra time and energy for security. SeBCDCP-ENC has a better performance than SeBCDCP-MAC. In the setup phase, the energy consumption is decided by the BS's reaction time: once the node has finished processing the incoming 2nd and (CH) or 3rd message (CM), it goes to sleep and waits for the data communication phase to start. The faster the response of the BS, the better the implementation performance.

10.5 Comparison of Heinzelman's Model Estimation Results and Empirical Measurement Results

Using Heinzelman's energy consumption estimation model and the packet sizes adopted by the implementations of the BCDCP, SeBCDCP-MAC, and SeBCDCP-ENC protocols, the energy consumption comparison is obtained. Table 10.5 shows a significant difference between the empirical measurement and estimation results. Heinzelman's estimation concluded both CH and CM have a steady increased energy consumption ratio of the secured protocols: SeBCDCP-MAC requires 27.27% and SeBCDCP-ENC requires 48.48% than original BCDCP. However, the empirical measurement results showed the opposite: SeBCDCP-MAC has more energy consumption than SeBCDCP-ENC and BCDCP. Since the empirical measurement results are more convincing, Heinzelman's model may introduce errors to the estimated results.

Table 10.5 Energy Consumption Comparison Between Estimated and Empirical Results

Protocol	CM Setup Phase		CH Setup Phase	
	Empirical	Estimation	Empirical	Estimation
BCDCP	100.00%	100.00%	100.00%	100.00%
SeBCDCP-ENC	114.12%	148.48%	131.72%	148.48%
SeBCDCP-MAC	134.13%	127.27%	168.84%	127.27%
Protocol	CM Data Com. Phase		CH Data Com. Phase	
	Empirical	Estimation	Empirical	Estimation
BCDCP	100.00%	100.00%	100.00%	100.00%
SeBCDCP-ENC	148.42%	148.48%	132.67%	148.48%
SeBCDCP-MAC	151.47%	127.27%	133.15%	127.27%

Part 3.

Energy Consumption Estimation Model

11. Requirement of New Energy Consumption Estimation Model

In Chapter 9 and 10, I have found significant inconsistencies between the existing estimation models and the empirical measurements taken from the implementations. In this chapter, I will analyze the traditional estimation models' inadequacies and investigate the feasibility of a more comprehensive energy consumption estimation model for WSN routing protocols.

11.1 Inadequacies of Xiao's Model

As discussed in section

Xiao's Model neglected the processing cycles for MOV or MOVW operations. Same as Xiao's assumption, MICAZ mote's ATmega128 microcontroller requires 1 hold of its processing time to perform a simple AND, OR, or SHIFT operation [Atm11]. However, one MOV or MOVW operation should be executed along with each AND or OR operation to store the computed value, which also takes 1 hold. SHIFT operates within the same register, so it does not requires this extra hold. Within a single block AES encryption, AND and OR operations are executed 2988 times, and SHIFT operates 3180 times. Thus the MOV or MOVW operation also needs to be carried out for 2988 times, which is ignored by Xiao.

Thus, in Xiao's model, around 50% of operation time was missing. This ratio rises to around 80% in single block AES-128 decryption, as AND and OR operate 10404 times, while SHIFT is only executed for 2028 times.

11.2 Inadequacies of Heinzelman's Model

Heinzelman's radio energy consumption model presented in [HCB00] also has its inadequacies: first, the radio model does not contain the energy consumption on the microcontroller side. In line with the MICAZ's specifications, the microcontroller requires approximately 40% of the radio transceiver's current draw, which should not be neglected; second, in the real world, the radio transceiver transmits the data with pre-defined signal strengths, which cannot be adjusted according to the communication distance [Atm03] [Chi04].

Last, as the radio transceiver usually has a relatively steady current draw in the receiving mode [Atm03] [Chi04], the energy consumption for a radio transceiver to receive data is decided by the duration for which it is active. For receiving data, the transceiver normally works as shown in Figure 10.5. The software can either setup a timer or use an event handler to wake up or knock out the radio transceiver [LNR05]. As the radio transceiver is controlled by software, the energy consumption needed to receive data is decided by not only the number of bytes to be transmitted or received, but also the time for the bytes to be processed and the software design of the radio transceiver on-off strategy (for example, the WSN nodes have to keep its radio transceiver active for a software-regulated time, to wait for the incoming packets).

In general, the theoretical time or energy consumption estimation models are "not good enough" for practical evaluation. The energy consumption on the

processor or radio transceiver (either in the transmission mode or in the reception mode) relates most to the software. Thus, using Heinzelman's radio model to estimate the energy consumption, which has been adopted by [OFV07] [MMB05] [You04] [MRK05], may not produce sufficient overall results, or at least it was not able to in our test.

11.3 Requirement for More Precise WSN Energy Consumption Estimation Model

To estimate the energy consumption of a WSN routing protocol in a more comprehensive and precise way, one can implement the protocol and experimentally measure the energy consumption of the WSN nodes. The solution can verify the routing protocol's applicability in regards to the specific sensor hardware. However, the design and implementation is a time and money consuming procedure as not only the necessary sensor hardware and measurement equipment is costly, but the test setup, debugging, and actual measurement require time. Therefore, in my opinion, to evaluate a WSN routing protocol's energy efficiency, theoretical models are more feasible.

However, two popular models, Xiao's processor time estimation method and Heinzelman's radio energy dissipation estimation model have their disadvantages: they neglected some modules' energy consumption. Both models cannot reflect the power consumption situations of real-world WSN nodes like MICAZ nodes, as they all contain processor and wireless communication modules [ASS02] [RMG02]. More importantly, even while estimating the energy consumption of the modules they are not neglected, the estimation results show relative big errors: Xiao's computational results, although based on the detailed operation number of AES algorithm, have over 33% estimation error than the

11. Requirement of New Energy Consumption Estimation Model

measurement results on MICAZ; Heinzelman's model, on the other hand, produces inconsistent estimation results with the measured energy consumption comparison.

Since the popular estimation models have their inadequacies, new energy consumption estimation should be developed to improve this situation.

In my opinion, the new estimation model should:

- Have higher accuracy than existing models.
- Target at least one type of popular WSN node hardware, and may be applied on multiple hardware kinds. Currently, a number of different WSN node hardware is available, each with different hardware specifications. Designing a generic estimation model for all of them may not be feasible. For example, some WSN node like IntelIMote2 contains powerful processors with automatic clock speed adjustment capability [Cro12d], while MICAZ's ATmega128 8-bit processor can only operate on full clock speed (8MHz) or sleep mode. It is unlikely to discover a single equation to estimate the totally two different processors. Thus, the new model should focus on estimating the energy consumption for one popular WSN node kind first, and, if possible, then extend to multiple hardware kinds in the future.
- Cover the energy consumption on both processor module and radio communication module. Heinzelman's model does not cover the energy estimation of processor module; Xiao's model is not designed for evaluating wireless communication model's energy consumptions. The new model should be able to estimate both modules. The sensing task is often irrelevant to the routing. The new model can neglect the sensor module's energy consumption estimation.

- Be able to estimate secured WSN routing protocol's energy consumption. Since more routing protocols are involved in security protection, the new model should be able to estimate the energy cost of the security operations like data encryption. Xiao's model, although containing some inadequacies, shows a good direction to estimate security protections' energy consumption.

11.4 Determine the Feasibility of Designing WSN Node Energy Consumption Estimation Model

To determine the feasibility of designing a comprehensive WSN routing protocol energy consumption estimation model, one important prerequisite condition is that both the WSN node's microcontroller and radio transceiver should have a relatively steady power draw in the predefined working modes, like sleep mode or active mode. This feature is possessed by many categories of sensor hardware [Chi04] [Cro12a] [Cro12b] [Cro12c]. Furthermore, a linear relationship should exist between the number of data bytes to be processed and the time consumed. Numerous radio transceivers [Chi04] [Tex07] [Atm09] complying with 2.4GHz Zigbee Standard have a relatively steady data rate of 250kbit/s [BPV07]. In this section, I will study the characteristics of a specific WSN node, MICAZ, and combine the experience gained in the implementations in order to explore the feasibility of designing a comprehensive estimation model to evaluate the WSN node's energy consumption while running a specific application.

11.4.1 Studying MICAZ's Hardware Working Characteristics

According to MICAZ's hardware specification [Cro12a] and WSN node's current draw recorded by Agilent 66321D (Cf. Figure 11.1, Figure 11.2, and Figure 11.3), we have concluded that MICAZ has three particular modes:

11. Requirement of New Energy Consumption Estimation Model

- Soft Sleep Mode: an approximately 3mA current draw will be consumed to support MICAZ's minimum functionality.
- Microcontroller Active Mode: an approximately 7.4mA current draw will be consumed in this mode (shown as Figure 11.1).

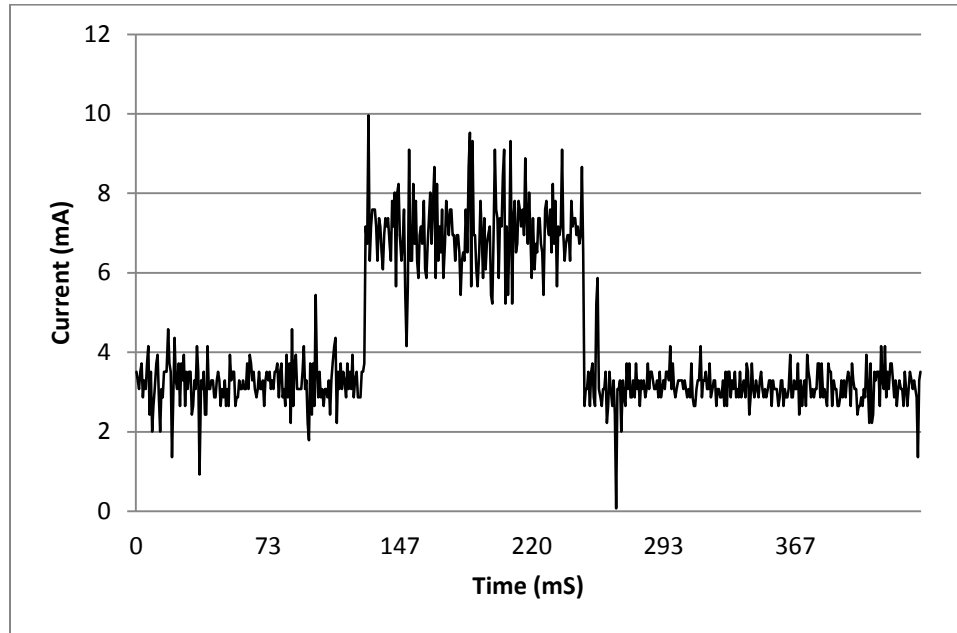


Figure 11.1 Sample of MICAZ Current Draw Waveform (Radio Off)

- Radio Transceiver Transmission Mode: both microcontroller and radio transceiver will be powered in this mode. The radio transceiver will consume an extra 17mA current for 0dBm transmission power (shown as Figure 11.2) and 11mA for -5dBm power.
- Radio Transceiver Reception or Listening Mode: same as in transmission mode, both microcontroller and radio transceiver will be powered in reception or listening mode. In this mode, an extra 19.7mA current draw will be consumed by the radio transceiver (shown as Figure 11.3).

ATmega128L microcontroller in MICAZ is not capable of adjusting its clock speed to minimize the power consumption. Thus it can only work in two modes: sleep mode and full clock speed. Thus, the microcontroller immediately goes into sleep mode after finishing all its tasks in the queue [LMP05]; if the actual

11. Requirement of New Energy Consumption Estimation Model

computation time can be deduced, the power consumption on the microcontroller can be known. Therefore, this also applies to the CC2420 radio transceiver.

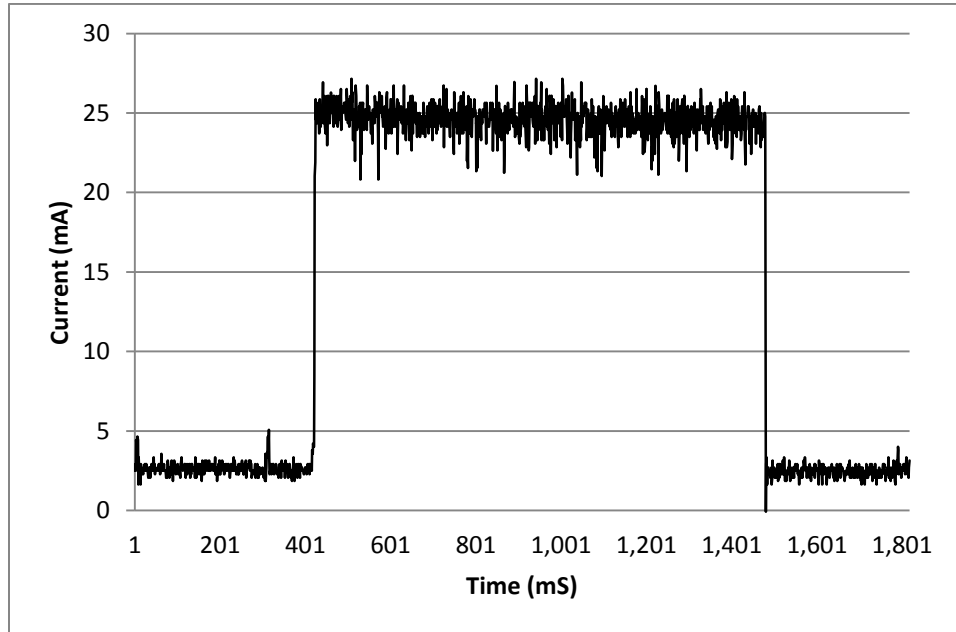


Figure 11.2 Sample of MICAZ Current Draw Waveform (Transmission Mode)

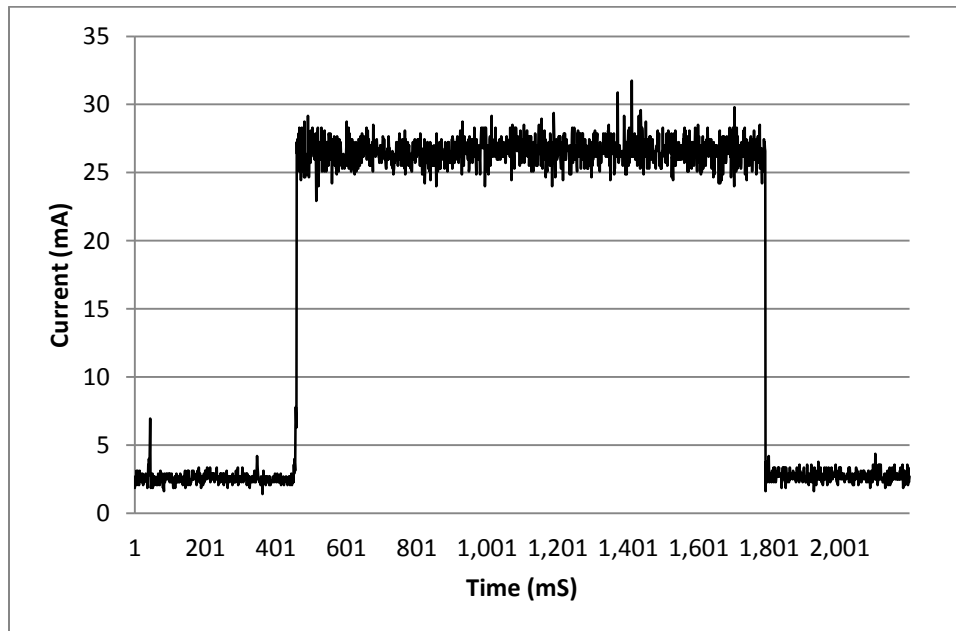


Figure 11.3 Sample of MICAZ Current Draw Waveform (Reception or Listen Mode)

12. A Novel Wireless Sensor Network Routing Protocol Energy Consumption Estimation Model (PPECEM)

In this chapter, a new WSN Practical Routing Protocol Energy Consumption Estimation Model (PPECEM) is introduced. The proposed PPECEM aims to improve the inadequacies of both Heinzelman's and Xiao's models and provide a new comprehensive energy consumption estimation model for WSN routing protocols running on MICAZ motes.

12.1 Relationship between PPECEM and Heinzelman's and Xiao's Models

The proposed model (PPECEM) is inspired by Heinzelman's and Xiao's models. In this section, I am going to describe my way to improve the inadequacies of the both models in PPECEM.

12.1.1 Improve Xiao's Model

As discussed in Section 11.1, the computational results presented in [XCS06] are not adequate: Xiao's model neglected 50% of processing time for encrypting a block of data, and 80% of time for decrypting a block. Thus, Xiao's computation results are adopted in PPECEM, but is then multiplied a factor of 1.5 for encryption, and 1.8 for decryption to fill what [XCS06] has been missed.

12.1.2 Estimating Energy Consumption on Radio Transceiver

Heinzelman's model has some limitations: first, it takes the transmission distance as one of the key factors to decide the output energy consumption; second, Heinzelman's model requires the knowledge of the amplifier parameters and the threshold distance to compute exact energy consumption. These values are hard to decide as they are not presented in many WSN node hardware specifications [Cro12a] [Cro12b] [Cro12c] [[Cro12d].

After going through the implementations of a few WSN routing protocols, I found the WSN nodes (like MICAZ) may have no location and distance knowledge to the destination nodes, and the radio transceivers have only minimum capabilities to adjust its output. Thus, the transmission distance should not be one of the factors to decide the radio transceiver's energy consumption. Moreover, the amplifier parameters and threshold distance are often not included in the hardware specifications rate [Cro12a] [Cro12b] [Cro12c].

Since one of PPECEM's design objectives is the targeting WSN hardware specifications, I start to check if there are any specs useful for the energy consumption estimation, and I found them. They are power source voltage

supply, microcontroller's current draw, radio transceiver's current draw in the transmission, reception, and listening modes, and the transmission data. Using these known factors, it is possible to deduce the per byte energy consumption on the target WSN node's radio transceiver.

12.2 PPECEM's General Equation

In general, the proposed PPECEM focuses on the WSN node's components active time. As we discussed in Chapter 2, a typical WSN node is composed of three components that consume power: the sensor module, the processor module, and the wireless communication module. Since one of the main purposes of PPECEM is aims only the WSN routing protocol and we can always assume the sensor module consumes same amount of energy in two WSN routing protocols to be compared with each other. Hence, PPECEM only includes the energy estimation of processor module and the wireless communication module.

PPECEM can be presented in brief as:

$$Q_{Overall} = Q_{CPU} + Q_{RadioTrans} + Q_{RadioRcv} \quad (\text{Eq. 12.1})$$

where $Q_{X(Y)}$ denotes the energy consumption of the device or device X or operation Y . The devices include the processor module's computation (Q_{CPU}), the wireless communication module's message transmission ($Q_{RadioTrans}$), and the wireless communication module's message reception ($Q_{RadioRcv}$). I will introduce how these computations are deduced in the following sections.

12.3 Deducing the Processor Module's Active Time

Normally, the security computation requires a lot more time than the routing computation [LKS10]. Therefore, PPECCEM only computes the processor module's energy consumption for security computations such as block cipher encryption, as a result the one for the unsecured WSN routing protocols is ignored. We have:

$$Q_{CPU} = P_{CPU} * T_{CPU} = P_{CPU} * (B_{Enc} * TB_{Enc} + B_{Dec} * TB_{Dec} + B_{Mac} * TB_{Mac} + T_{RadioActive}) \quad (\text{Eq. 12.2})$$

PX in Eq. 12.2 represents the power of device X. TX means the computation time of device X and TBY denotes the per-byte time consumed for doing operation Y. BY indicates the amount of bytes to be computed by operation Y. Enc, Dec, and Mac denote the encryption, decryption, and MAC digest generation operations, respectively. TRadioActive represents the radio transceiver's active time, as the processor module remains active while the radio chip is turned on.

P_{CPU} can be found in the WSN node hardware's specifications. For instance, MICAZ's $P_{CPU} = 8mA * 3V = 24mW$. T_{Enc} and T_{Dec} are derived from Xiao's work on [XCS06]. In Xiao's work, a 100 MIPS processor requires $59\mu s$ to encrypt a block of data using a 128-bit key, and a $123\mu s$ for decryption. As discussed in Section 12.1.1, I multiply a factor 1.5 for encryption, and 1.8 for decryption to fill what [XCS06] has been missed. After applying the factors, we have:

$$TB_{Enc} = \frac{59\mu s * 1.5 * 100}{N - MIPS * 16} \quad (\text{Eq. 12.3})$$

$$TB_{Dec} = \frac{123\mu s * 1.8 * 100}{N - MIPS * 16} \quad (\text{Eq. 12.4})$$

MICAZ mote hardware is equipped with an 8-MIPS processor module. Thus:

$$TB_{Enc} = \frac{59\mu s * 1.5 * 100}{8 * 16} = 69.14 \mu s$$

$$TB_{Dec} = \frac{123\mu s * 1.5 * 100}{8 * 16} = 144.14 \mu s$$

TB_{Enc} can be used to deduce TB_{Mac} , if the target implementation adopts AES-CBC-MAC as its MAC algorithm. AES-CBC-MAC divides the message into fragments. Each block has an equal size of X bytes ($X \in \{2, 4, 8, 16\}$). In case of the last fragment is less than X bytes, it will be padded before it is processed. According to CBC-MAC's definition (Cf. Figure 5.4), each fragment has to be encrypted. I further neglect the processing time for the XOR operation in AES-CBC-MAC operation (XORing two blocks data takes about 32 processor cycles, while single block encryption takes 8980 cycles, neglecting less than 1% of time should not affect the overall energy consumption much). Thus:

$$TB_{Mac} = TB_{Enc} * 16 / X \quad (\text{Eq. 12.5})$$

For the circumstance of $X = 8$, which was what I adopted in the SeBCDCP-MAC implementation, $TB_{Mac} = TB_{Enc} * 2$.

12.4 Deducing the Wireless Communication Module's Transmission and Reception Energy Consumption

Many WSN nodes have relative steady transmission and reception data rates [Cro12a] [Cro12b] [Cro12c], thus per-byte transmission time should be stable. According to MICAZ datasheet, the radio transceiver have roughly same energy consumption on receiving and listening mode, thus their energy consumption computation can be combined. However, according to my experimental results,

12. A Novel WSN Protocol Energy Consumption Estimation Model (PPECEM)

some wireless communication modules, like CC2420, require processing to be ready for packet transmission, which also consumes energy and should be taken into consideration. Therefore, there is:

$$Q_{RadioTrans} = P_{RadioTrans} * (K_{Trans} * B_{Trans} + T_{Startup}) \quad (\text{Eq. 12.6})$$

$$Q_{RadioRcv} = P_{RadioRcv} * (K_{Rcv} * B_{Rcv} + T_{Listen}) \quad (\text{Eq. 12.7})$$

With the same idea as deducing P_{CPU} , we can deduce $P_{RadioTrans}$ and $P_{RadioRcv}$ according to the WSN node's hardware specification. For MICAZ, they are:

$$P_{RadioTrans} = 17.4mA * 3V = 52.5mW \text{ (for 0dB)}$$

$$P_{RadioRcv} = 19mA * 3V = 57mW$$

K_{Trans} and K_{Rcv} are the factors related to the time duration for the sensor to send or receive a byte, plus the time for the sensor to read/write a byte in/from the radio chip. I assume these two factors are equal. The transmission data rate can be found in the WSN node's hardware specification. In MICAZ, they are 40 μ s/byte. The MICAZ's radio chip read-write time is 20 μ s/byte [ZDC11] [HNL07] [DAB08]. Thus for MICAZ mote hardware, $K_{Trans} = K_{Rcv} = 60\mu$ s/byte.

$T_{Startup}$ denote the processing time for the wireless communication module to be ready for transmission. TinyOS may introduce a delay between the program launching sending command and when the packet is actually sent successfully. This delay time can be obtained only via experiment. Currently I have confirmed this value in MICAZ's CC2420 is 6.14ms per sending.

T_{Listen} indicates the wireless communication module's waiting time between two transmitting and / or receiving tasks. It is always assumed the receptor WSN

12. A Novel WSN Protocol Energy Consumption Estimation Model (PPECEM)

node or the BS will immediately react to the sent messages. For example, soon after receiving an encrypted message, the receptor WSN node or the BS will start the decryption of the message. After the decryption is done, the node will start encrypting (if needed) the message to be responded to, and sent out. Hence T_{Listen} should be equal to the receptor node's messages receiving and sending time, plus the time for security.

In general, the proposed new model can be presented as follows:

$$Q_{Overall} = P_{CPU} * (B_{Enc} * \frac{59\mu s * 1.5 * 100}{N - MIPS * 16} + B_{Dec} * \frac{123\mu s * 1.8 * 100}{N - MIPS * 16}) \\ + P_{RadioTrans} * (K_{Trans} * B_{Trans} + T_{Startup}) + P_{RadioRcv} * (K_{Rcv} * B_{Rcv} + T_{Listen}) \quad (Eq. 12.8)$$

13. Evaluate PPECEM

In this chapter, I am going to evaluate PPECEM's precision in relation to estimating the WSN routing protocol's energy consumption. The referential energy consumption value is obtained by the energy consumption measurement of the routing protocol's practical implementations.

I will first evaluate the model's energy consumption computation on AES-128 single block encryption and decryption, and compare this against the practical measurement results (published on [ZDC11]). In the next sections, I compute the energy consumption on BCDTCP, SeBCDTCP-ENC, and SeBCDTCP-MAC protocols, and evaluate the difference between the practical measurement results.

13.1 AES-128 Encryption and Decryption Comparison

Since software AES implementation is not involved in the radio communication, the energy consumption is done only by the processor module. For MICAZ's 8-bit ATmega128L processor, there is:

$$Q_{AES-Encryption} = P_{CPU} * (B_{Enc} * TB_{Enc}) = 24mW * (16 * \frac{59\mu s * 1.5 * 100}{8 * 16}) = 26.42 \mu J$$

$$Q_{AES-Decryption} = P_{CPU} * (B_{Dec} * TB_{Dec}) = 24mW * (16 * \frac{123\mu s * 1.8 * 100}{8 * 16}) = 66.44 \mu J$$

Compared against the measured energy consumption value, the accuracies of the model's estimation results, including the results evaluated using Xiao's model, are listed in Table 13.1. In the table, the percentage values within the brackets are the estimation error compared against the empirical measurement results. In this comparison, the errors of the results computed using PPECEM are less than 14% than the measurement results, and is more than 50% more accurate of the results computed using Xiao's model.

Table 13.1 Software-AES Estimation and Empirical Measurement Results Comparison

	Measured (μJ)	Xiao's Estim. (μJ)	Proposed Estim. (μJ)
AES Encryption.	23.60 (0%)	17.65 (-25.18%)	26.42 (11.97%)
AES Decryption.	58.60 (0%)	36.90 (-37.03%)	66.44 (13.39%)

13.2 BCDCP Setup Phase Energy Consumption Estimation

➤ Estimation of Power Consumption of CM

BCDCP is not involved in data encryption. However, the processor module has to remain active while the radio transceiver is being turned on. Thus, in BCDCP the processor module and the wireless communication module have the same active time.

Based on the discussion of BCDCP implementation, the CM in the BCDCP sends and receives one packet (17 bytes) in the setup phase. Thus its transmission time should be equal to the 17 bytes' data transmission time and can be deduced from Eq. 12.6. However, the reception time estimation is slightly complex. In my implementation scheme design (Cf. Figure 10.2), before it receives the 3rd

13. Evaluate PPECEM

message, the CM needs to wait for BS's responses. BS first receives two 1st messages from the CH and the CM, and then sends 2nd message to the CH before the 3rd message is sent (in addition the BS is a MICAZ mote as well, and it has to transmit the message sequentially). According to Eq. 12.7, the computation process of the BCDP CM in the setup phase is computed as follows:

$$Q_{Radio} = P_{RadioTrans} * (K_{Trans} * B_{Trans} + T_{Startup}) + P_{RadioRcv} * (K_{Rcv} * B_{Rcv} + T_{Listen})$$

Where:

$$B_{Trans} = 1^{st} \text{ message size} = 33 \text{ bytes}$$

$$T_{Startup} = 1^{st} \text{ message startup time} = 6.14 \text{ ms}$$

$$B_{Rcv} = 2^{nd} \text{ message size} = 33 \text{ bytes}$$

$$T_{Listen} = \text{BS respond time} = \text{two } 1^{st} \text{ message reception time} + 3^{rd} \text{ message startup time} + 3^{rd} \text{ message send time}$$

$$= 2 * K_{Rcv} * 33 + 6.14 + K_{Trans} * 33$$

$$\text{For MICAZ mote, } K_{Trans} = K_{Rcv} = 0.04 + 0.02 = 0.06 \text{ ms/byte}$$

Thus

$$\begin{aligned} Q_{Radio} &= 52.5 * (0.06 * 33 + 6.14) + 57 * (0.06 * (33 * 4 + 6.14)) \\ &= 52.5 \text{ mW} * 8.12 \text{ ms} + 57 \text{ mW} * (1.98 + 12.08) \text{ ms} \\ &= 1.23 \text{ mJ} \end{aligned}$$

$$\begin{aligned} Q_{CPU} &= P_{CPU} * T_{CPU} \\ &= 24 \text{ mW} * (8.12 + 1.98 + 12.08) \text{ ms} \\ &= 0.53 \text{ mJ} \end{aligned}$$

$$\begin{aligned} Q_{Overall} &= Q_{CPU} + Q_{RadioTrans} + Q_{RadioRcv} \\ &= 1.76 \text{ mJ} \end{aligned}$$

➤ **Estimation of Power Consumption of CH**

BCDCP's CH has a different radio reception time than the CM, as it needs to receive an extra message from the BS in the setup phase. Hence, the CH's waiting time should cover the BS's 1st message (twice including CM's 1st message reception) reception time, and 2nd message sending time in the setup phase, and 4th message's sending time and receiving time in the data communication phase.

With the same computation procedure of BCDCP CM, the energy consumption on BCDCP CH in the setup phase can be computed as:

$$Q_{Radio} = P_{RadioTrans} * (K_{Trans} * B_{Trans} + T_{Startup}) + P_{RadioRcv} * (K_{Rcv} * B_{Rcv} + T_{Listen})$$

Where

$$B_{Trans} = 1^{st} \text{ message size} = 33 \text{ bytes}$$

$$T_{Startup} = 6.14 \text{ ms}$$

$$B_{Rcv} = 2^{nd} \text{ message size} = 33 \text{ bytes}$$

$$\begin{aligned} T_{Listen} &= \text{BS respond time} = \text{two } 1^{st} \text{ message reception time} + 3^{rd} \text{ message startup} \\ &\text{time} + 3^{rd} \text{ message send time} + 2^{nd} \text{ message startup time} + 2^{nd} \text{ message send time} \\ &= 2 * K_{Rcv} * 33 + 6.14 * 2 + K_{Trans} * 33 * 2 \end{aligned}$$

Thus

$$\begin{aligned} Q_{Radio} &= 52.5 * (0.06 * 33 + 6.14) + 57 * (0.06 * 33 * 5 + 6.14 * 2) \\ &= 1.69 \text{ mJ} \end{aligned}$$

$$\begin{aligned} Q_{CPU} &= P_{CPU} * T_{CPU} = 24 \text{ mW} * 32.66 \text{ ms} \\ &= 0.72 \text{ mJ} \end{aligned}$$

$$Q_{Overall} = Q_{CPU} + Q_{RadioTrans} + Q_{RadioRcv} = 2.41 \text{ mJ}$$

13.3 SeBCDCP-ENC Setup Phase Energy Consumption Computation

In [ZDC11] the table-lookup version of AES requires only 80.2% of encryption time of the original AES and only 41.99% of decryption time of original AES. Lee et al. also stated the table-lookup version of AES require 8980 and 9120 processor cycles to encrypt and decrypt a block of data, respectively [LLS09]. Two publications show a similar results, thus we can take the same ratios from the estimated original AES times. Thus we obtain $TB_{Enc} = 56.42\mu\text{s}/\text{byte}$, and $TB_{Dec} = 60.52\mu\text{s}/\text{byte}$ for the optimized table lookup AES.

➤ Estimation of Power Consumption of CM

In my implementation designs, SeBCDCP-ENC is involved in the symmetric data encryption and decryption. I utilized the table-lookup AES implementations to encrypt and decrypt data. Thus the processor module's active time should also cover the processing time for encrypting and decrypting blocks. In our implementation, the CM and BS in the setup phase need to encrypt and decrypt a few blocks. Apart from that, the computation procedure of SeBCDCP-ENC should be the same as the BCDCP.

$$Q_{Radio} = P_{RadioTrans} * (K_{Trans} * B_{Trans} + T_{Startup}) + P_{RadioRcv} * (K_{Rcv} * B_{Rcv} + T_{Listen})$$

Where

$$B_{Trans} = 1^{\text{st}} \text{ message size} = 49 \text{ bytes}$$

$$T_{Startup} = 6.14 \text{ ms}$$

$$B_{Rcv} = 2^{\text{nd}} \text{ message size} = 49 \text{ bytes}$$

13. Evaluate PPECEM

$$\begin{aligned} T_{\text{Listen}} &= \text{BS respond time} = \text{two 1}^{\text{st}} \text{ message reception time} + \text{two 1}^{\text{st}} \text{ message} \\ &\text{decryption time} + \text{3}^{\text{rd}} \text{ message encryption time} + \text{3}^{\text{rd}} \text{ message startup time} + \text{3}^{\text{rd}} \\ &\text{message send time} \\ &= 2 * 49 * K_{\text{Rcv}} + TB_{\text{Enc}} * 16 * 2 + TB_{\text{Dec}} * 32 + 6.14 + K_{\text{Trans}} * 49 \end{aligned}$$

$$\text{Thus } Q_{\text{Radio}} = 1.71 \text{ mJ}$$

$$Q_{\text{CPU}} = P_{\text{CPU}} * T_{\text{CPU}} = P_{\text{CPU}} * (B_{\text{Enc}} * TB_{\text{Enc}} + B_{\text{Dec}} * TB_{\text{Dec}} + T_{\text{RadioActive}})$$

Where

$$B_{\text{Enc}} = \text{1}^{\text{st}} \text{ message payload} = 16 \text{ bytes}$$

$$B_{\text{Dec}} = \text{3}^{\text{rd}} \text{ message payload} = 32 \text{ bytes}$$

$$T_{\text{RadioActive}} = T_{\text{Trans}} + T_{\text{Rcv}} = 33.57\text{ms}$$

Thus

$$\begin{aligned} Q_{\text{CPU}} &= 24 * (16 * 0.056 + 32 * 0.061 + 33.57) \\ &= 0.81 \text{ mJ} \end{aligned}$$

$$\begin{aligned} Q_{\text{Overall}} &= Q_{\text{CPU}} + Q_{\text{RadioTrans}} + Q_{\text{RadioRcv}} \\ &= 2.52 \text{ mJ} \end{aligned}$$

➤ Estimation of Power Consumption of CH

Same as with the SeBCDCP-ENC's CM energy consumption computation, the CH's processor module active time should be involved in the messages' encryption and decryption. The computation procedure is shown as follows:

$$Q_{\text{Radio}} = P_{\text{RadioTrans}} * (K_{\text{Trans}} * B_{\text{Trans}} + T_{\text{Startup}}) + P_{\text{RadioRcv}} * (K_{\text{Rcv}} * B_{\text{Rcv}} + T_{\text{Listen}})$$

13. Evaluate PPECCEM

Where

$$B_{Trans} = 1^{st} \text{ message size} = 49 \text{ bytes}$$

$$T_{Startup} = 6.14 \text{ ms}$$

$$B_{Rcv} = 2^{nd} \text{ message size} = 49 \text{ bytes}$$

$$T_{Listen} = \text{BS respond time} = \text{two } 1^{st} \text{ message reception time} + \text{two } 1^{st} \text{ message decryption time} + 3^{rd} \text{ message encryption time} + 3^{rd} \text{ message startup time} + 3^{rd} \text{ message send time} + 2^{nd} \text{ message encryption time} + 2^{nd} \text{ message startup time} + 2^{nd} \text{ message send time}$$

$$= 2 * K_{Rcv} * 49 + TB_{Dec} * 16 * 2 + 32 * TB_{Enc} + 6.14 + 49 * K_{Trans} + 48 * TB_{Enc} + 6.14 + 49 * K_{Trans}$$

$$\text{Thus } Q_{Radio} = 2.35 \text{ mJ}$$

$$Q_{CPU} = P_{CPU} * T_{CPU} = P_{CPU} * (B_{Enc} * TB_{Enc} + B_{Dec} * TB_{Dec} + T_{RadioActive})$$

Where

$$B_{Enc} = 1^{st} \text{ message payload} = 16 \text{ bytes}$$

$$B_{Dec} = 2^{nd} \text{ message payload} = 48 \text{ bytes}$$

$$T_{RadioActive} = T_{Trans} + T_{Rcv} = 42.07 \text{ ms}$$

Thus

$$Q_{CPU} = 24 * (16 * 0.056 + 48 * 0.061 + 42.07) = 1.10 \text{ mJ}$$

$$Q_{Overall} = Q_{CPU} + Q_{RadioTrans} + Q_{RadioRcv} = 2.64 + 1.34 = 3.46 \text{ mJ}$$

13.4 SeBCDCP-MAC Setup Phase Energy Consumption Computation

The SeBCDCP-MAC's energy consumption estimation has similar procedure with SeBCDCP-ENC. For SeBCDCP-MAC implementation, table-lookup AES algorithm is used to encrypt / decrypt cluster keys and generate messages' MAC digests. I will describe its setup phase's energy consumption estimation procedure in this sub-section. In Section 13.3, I have concluded AES encryption time takes 0.056ms/byte. In the SeBCDCP-MAC implementation, the MAC fragment size is 8 bytes. Thus

$$TB_{Mac} = TB_{Enc} * 16 / 8 = 0.056 * 2 = 0.112 \text{ ms/byte}$$

➤ Estimation of Power Consumption of CM

$$Q_{Radio} = P_{RadioTrans} * (K_{Trans} * B_{Trans} + T_{Startup}) + P_{RadioRcv} * (K_{Rcv} * B_{Rcv} + T_{Listen})$$

Where

$$B_{Trans} = 1^{st} \text{ message size} = 42 \text{ bytes}$$

$$T_{Startup} = 6.14 \text{ ms}$$

$$B_{Rcv} = 2^{nd} \text{ message size} = 42 \text{ bytes}$$

T_{Listen} = BS respond time = two 1st message reception time + two 1st message MAC digest generation time + 16-byte cluster key encryption time + 3rd message digest gen. time + 3rd message startup time + 3rd message send time

$$= 2 * 42 * K_{Rcv} + TB_{Mac} * 2 * 16 + TB_{Enc} * 16 + TB_{Mac} * 32 + 6.14 + K_{Trans} * 42$$

$$\text{Thus } Q_{Radio} = 1.84 \text{ mJ}$$

$$Q_{CPU} = P_{CPU} * T_{CPU} = P_{CPU} * (B_{Enc} * TB_{Enc} + B_{Mac} * TB_{Mac} + T_{RadioActive})$$

Where

$$B_{Enc} = 1 \text{ cluster key size} = 16 \text{ bytes}$$

$$B_{Mac} = 3^{\text{rd}} \text{ message payload} = 32 \text{ bytes}$$

$$T_{RadioActive} = T_{Trans} + T_{Rcv} = 32.94\text{ms}$$

Thus

$$\begin{aligned} Q_{CPU} &= 24 * (16 * 0.056 + 32 * 0.112 + 32.94) \\ &= 0.97 \text{ mJ} \end{aligned}$$

$$\begin{aligned} Q_{Overall} &= Q_{CPU} + Q_{RadioTrans} + Q_{RadioRcv} \\ &= 2.81 \text{ mJ} \end{aligned}$$

➤ **Estimation of Power Consumption of CH**

$$Q_{Radio} = P_{RadioTrans} * (K_{Trans} * B_{Trans} + T_{Startup}) + P_{RadioRcv} * (K_{Rcv} * B_{Rcv} + T_{Listen})$$

Where

$$B_{Trans} = 1^{\text{st}} \text{ message size} = 42 \text{ bytes}$$

$$T_{Startup} = 6.14 \text{ ms}$$

$$B_{Rcv} = 2^{\text{nd}} \text{ message size} = 42 \text{ bytes}$$

$$\begin{aligned} T_{Listen} &= \text{BS respond time} = \text{two } 1^{\text{st}} \text{ message reception time} + \text{two } 1^{\text{st}} \text{ message} \\ &\text{digest gen. time} + \text{cluster key enc. time} + 3^{\text{rd}} \text{ message digest gen. time} + 3^{\text{rd}} \\ &\text{message startup time} + 3^{\text{rd}} \text{ message send time} + \text{cluster key enc. time} + 2^{\text{nd}} \\ &\text{message digest gen. time} + 2^{\text{nd}} \text{ message startup time} + 2^{\text{nd}} \text{ message send time} \\ &= 2 * 42 * K_{Rcv} + TB_{Mac} * 2 * 16 + TB_{Enc} * 16 + TB_{Mac} * 32 + 6.14 + K_{Trans} * 42 + TB_{Enc} * \\ &16 + TB_{Mac} * 48 + 6.14 + K_{Trans} * 42 \end{aligned}$$

$$\text{Thus } Q_{Radio} = 2.73 \text{ mJ}$$

13. Evaluate PPECCEM

$$Q_{CPU} = P_{CPU} * T_{CPU} = P_{CPU} * (B_{Enc} * TB_{Enc} + B_{Mac} * TB_{Mac} + T_{RadioActive})$$

Where

$$B_{Enc} = 1^{st} \text{ message payload} = 16 \text{ bytes}$$

$$B_{Mac} = 2^{nd} \text{ message payload} = 48 \text{ bytes}$$

$$T_{RadioActive} = T_{Trans} + T_{Rcv} = 48.51 \text{ ms}$$

Thus

$$\begin{aligned} Q_{CPU} &= 24 * (16 * 0.056 + 48 * 0.061 + 42.07) \\ &= 1.36 \text{ mJ} \end{aligned}$$

$$Q_{Overall} = Q_{CPU} + Q_{RadioTrans} + Q_{RadioRcv} = 2.73 + 1.36 = 4.09 \text{ mJ}$$

13.5 Routing Protocols' Evaluation Accuracy

In general, PPECCEM's computation results are listed in Table 13.2. PPECCEM is expected to have less than 12% error in order to compute the actual energy consumption amount.

It is difficult to make a quantitative comparison between the proposed PPECCEM and the Heinzelman's model: due to the lack of the amplifier parameters and the threshold distance values, it is impossible to compute the protocols' exact energy consumption using Heinzelman's model. Thus, only the qualitative comparison between PPECCEM and Heinzelman's model is accessible. I reused the figures listed in Table 10.5, and added the ratios of PPECCEM's computation results, to Table 13.3. Again, the energy consumptions of CH and CM in BCDCP are set as ratio of 100%.

13. Evaluate PPECEM

From Table 13.3 we can see, While PPECEM's computed energy consumption ratios are in line with the empirical measurement results, and has closer ratios in 3 out 4 items. Although PPECEM's relative ratio error was bigger than Heinzelman's computation result in SeBCDCP-MAC CM's extra energy consumption error, Heinzelman's results were inconsistent with the empirical measurement results in the first place: the empirical results show SeBCDCP-MAC has more energy consumption than SeBCDCP-ENC, where Heinzelman said the opposite. Thus, PPECEM has in fact better qualitative results than Heinzelman model.

Table 13.2 PPECEM's Estimation Results

	Empirical (mJ)	Error (%)	PPECEM (mJ)	Est. Error (%)
BCDCP CM	1.99	0.00%	1.76	-11.56%
BCDCP CH	2.69	0.00%	2.41	-10.41%
SB-ENC CM	2.27	0.00%	2.52	11.01%
SB-ENC CH	3.55	0.00%	3.46	-2.54%
SB-MAC CM	2.67	0.00%	2.81	5.24%
SB-MAC CH	4.55	0.00%	4.09	-10.11%

Table 13.3 Ratio Comparison of Three Protocols

	Empirical	PPECEM		Heinzelman	
		Ratio	Diff.	Ratio	Diff.
BCDCP CM	100.00%	100.00%	0.00%	100.00%	0.00%
SB-ENC CM	114.07%	143.18%	29.11%	148.48%	34.41%
SB-MAC CM	134.17%	159.66%	25.49%	127.27%	-6.90%
BCDCP CH	100.00%	100.00%	0.00%	100.00%	0.00%
SB-ENC CH	131.97%	143.57%	11.60%	148.48%	16.51%
SB-MAC CH	168.84%	169.29%	0.45%	127.27%	-41.57%

In general, PPECEM has a better performance than Heinzelman's model on estimating energy consumption of WSN routing protocols, and is a reliable energy consumption computational model for WSN routing protocols.

14. Conclusion

The main objective of this research was to identify and evaluate the energy consumption of the routing protocols for WSN. A study of WSN routing protocols, both unsecured and secured, was undertaken that included discussing WSN characteristics and limitations, WSN hardware composition, energy consumption and also attacks against WSN and their countermeasures. Subsequently, a security analysis of the BCDP protocol was performed and several weaknesses in the protocol were revealed.

Two secured versions of the protocol, namely SeBCDCP-ENC and SeBCDCP-MAC, were proposed and their security was analyzed. To evaluate the cost of the added security, the relative energy consumption of the three protocols was estimated using Heinzelman's model. Additionally, all three protocols have been implemented on a sample WSN system and empirical energy consumption measurements were taken from the three protocol implementations. Comparison of the empirical measurements against the estimated values revealed a disparity between them: Heinzelman's model indicated SeBCDCP-ENC had higher energy consumption than SeBCDCP-MAC, whereas the measurement results showed the opposite was the case.

14. Conclusion

Also, the energy consumption of three different AES implementations was analysed using Xiao's estimation model and by empirical measurements. Again, the estimation model showed significant error: Xiao's estimation results showed a 50.10% and 83.67% differences in comparison to the actual measurement results. Overall, the comparisons against empirical measurements showed that existing energy consumption models are not accurate. After the detailed study of the models presented by Xiao and Heinzelman, I found both models had disadvantages and neglected some important factors. A more comprehensive energy consumption estimation model should exist to improve this situation.

A novel WSN routing protocol energy consumption estimation model (PPECEM) was proposed. The model consists of the energy evaluation of the two WSN node's major components, namely the microcontroller and radio transceiver modules. The model computed the energy consumption on microcontrollers during: security operations, memory read-write between the microcontroller and the radio transceiver, and radio transceivers action (such as sending, receiving, and listen mode, etc.). A few parameters were involved in the model and most of them can be obtained from the WSN node's hardware specifications.

Lastly, I used PPECEM to compute the MICAZ's energy consumption of software AES encryption and decryption and the of the protocols BCDCP, SeBCDCP-MAC and SeBCDCP-ENC. Comparison of the new model's result against existing estimation models shows that the new model is significantly closer to the measured values.

The future work of the research includes

A. Avail PPECEM to estimate the energy consumption for more WSN node hardware kinds. Currently PPECEM is only evaluated on MICAZ sensor hardware. In theory, PPECEM should fit the WSN node sensor hardware

- Having both processor modules and wireless communication modules.
- Both processor modules and wireless communication modules have finite number of states, each with relatively constant working efficiency and energy consumption. For example, the processor should have full power working state and sleep state; the wireless communication module have different transmission power states and sleep state, and steady data rate for each state (except for sleep state).
- The hardware specification manual should exist, and contains at least the following contents (or the specifications can be deduced to): processor speed (in Hz), data rate (in bps), power supply voltage (in V), and current draw value (in A) in every specified state.

Currently, not only MICAZ mote, both IRIS mote [Cro12b] and TelosB mote [Cro12c] fits the requirement. However, even a WSN node fits all these requirements, an experiment have to be carried out to obtain the wireless communication module's start-up time (Cf. Section 12.4). The future works will involve in obtaining the start-up time for IRIS and TelosB first, and the precision of PPECEM's computation results on these two WSN node kinds will then be evaluated. Same work will be done on more WSN node hardware afterwards.

B. Completing PPECEM to avail more security algorithms' energy estimation. The security algorithms PPECEM supports are AES-128 encryption / decryption, and AES-CBC-MAC digest generation. The future work of the research will be involved in supporting more security algorithms, like hashing functions, hash-based MAC, more block ciphers, and public key creation and verifications.

Reference

- [Agi07] Agilent, "66321D Mobile Comm DC Source w/ Battery Emulation, DVM Datasheet", Obtained through Internet: http://www.home.agilent.com/upload/cmc_upload/All/66300series_datasheet_Jan07.pdf?&cc=IE&lc=eng, 2007. (Accessed 16-07-2012)
- [AlK04] J.N. Al-Karaki and A.E. Kamal, "Routing Techniques in Wireless Sensor Networks: a Survey", *Wireless Communications, IEEE*, Vol.11(6), 2004, pp.6-28.
- [ASS02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: a Survey", *Computer Networks, ACM*, Vol. 38(4), 15 March 2002, pp. 393-422
- [Atm03] Atmel Corporation, "AVR Application Note 109: Self Programming", San Jose, California, 2003.
- [Atm09] Atmel Corporation, "AT86RF230 Datasheet", 2009. Obtained through Internet: <http://www.atmel.com/Images/doc5131.pdf> (Accessed 16-07-2012)
- [Atm11] Atmel Corporation, "ATmega128, ATmega128L Datasheet", Obtained through Internet: <http://www.atmel.com/Images/doc2467.pdf>
- [Bar64] P. Baran, "On Distributed Communications Networks", *Communication Systems, IEEE*, Vol.12 (1), March 1964, pp. 1-9.
- [BBF03] G. Bertoni, L. Breveglieri, P. Fragneto, M. Macchetti, and S. Marchesin, "Efficient Software Implementation of AES on 32-Bit Platforms," *Lecture Notes in Computer Science*, Vol. 2523, Springer, 2003, pp. 129-142.
- [BCK96] M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication", *Advances in Cryptology - Crypto 96 Proceedings, Lecture Notes in Computer Science* Vol. 1109, Springer-Verlag, 1996.

- [Bla05] P.E. Black, "Greedy Algorithm", Dictionary of Algorithms and Data Structures, NIST, 2005. Obtained through Internet: <http://xlinux.nist.gov/dads//HTML/greedyalgo.html> (Accessed 30-10-2012)
- [BPV07] P. Barontib, P. Pillaia, W.C. Vince, S. Chessab, A. Gottab, and Y. F. Hua, "Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and Zigbee Standards", Computer Communications, Vol.30(7), Elsevier, 2007, pp. 1655-1695.
- [BSP10] J. Borms, K. Steenhaut, K. H. Phung, and B. Lemmens, "A Traffic-Adaptive Multi-Channel MAC Protocol for Wireless Sensor Networks", Proceedings of the International Conference on Communications and Electronics, August, Nha Trang, Vietnam, 2010, pp. 11-13.
- [Bys11] L. K. Bysani, "A Survey on Selective Forwarding Attack in Wireless Sensor Networks", Proceedings of International Conference on Devices and Communications, IEEE, 2011
- [Chi04] Chipcon, "CC2420 datasheet", 2004. Obtained through Internet: <http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf> (Accessed 16-07-2012)
- [CMY10] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor Network Security: a Survey", Communications Surveys & Tutorials, IEEE, Vol.11(2), 2010, pp. 52-73.
- [CoP02] N. Courtois, J. Pieprzyk. "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations", Lecture Notes in Computer Science, Vol. 2501, pp. 267-287. Springer, 2002
- [CoR08] T. Coffey and R. Dojen, "A Formal Verification Centered Design Process for Security Protocols", Handbook of Research on Information Security and Assurance, Eds: Gupta, J.N.D. and Sharma, S.K., IGI Global, August 2008, pp. 165-178, ISBN 978-1-59904-855-0.
- [Cro12a] CrossBow, "MICAZ Datasheet", 2012. Obtained through Internet: http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf (Accessed 16-07-2012)

- [Cro12b] CrossBow, "IRIS Mote Datasheet", 2012. Obtained through Internet: http://www.dinesgroup.org/projects/images/pdf_files/iris_datasheet.pdf (Accessed 16-07-2012)
- [Cro12c] CrossBow, "TelosB Mote Datasheet", 2012. Obtained through Internet: http://www.willow.co.uk/TelosB_Datasheet.pdf (Accessed 16-07-2012)
- [Cro12d] CrossBow, "Imote2 WSN Node Datasheet", 2012, Obtained through Internet: http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/Imote_2_Datasheet.pdf (Accessed 24-10-2012)
- [DAB08] S. Diala, A. Ault, and S. Bagchi, "Optimizing AES for Embedded Devices And Wireless Sensor Networks", Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities, Innsbruck, March 2008.
- [DaR99] J. Daemen and V. Rijmen, "AES Proposal: Rijndael", 1999.
- [DLC08] R. Dojen, I. Lasc, and T. Coffey, "Establishing and Fixing a Freshness Flaw in a Key-Distribution and Authentication Protocol", 2008 IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 28-30, 2008
- [Dob96] H. Dobbertin, "The Status of MD5 after a Recent Attack", CryptoBytes, Vol. 2(2), RSA Labs, Summer 1996.
- [DoC05] R. Dojen and T. Coffey, "The Concept of Layered Proving Trees and its Application to the Automation of Security Protocol Verification", ACM Transactions on Information and System Security (TISSEC), ACM, 2005, pp. 287-311.
- [Dou02] J. R. Douceur, "The Sybil Attack", Peer-to-peer Systems, Vol. 2429, Springer-Verlag, 2002, pp. 251-260.
- [Dwo01] M. Dworkin, "Recommendation for Block Cipher Modes of Operation. Methods and Techniques", Computer Security, NIST Special Publication 800-38A, 2001.

- [Dwo05] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", NIST Special Publication 800-38B, 2005.
- [DXC06] X. Du, Y. Xiao, H. Chen, and Q. Wu, "Secure Cell Relay Routing Protocol for Sensor Networks: Research Articles", *Journal of Wireless Communications and Mobile Computing - Wireless Network Security*, Vol.6 (3), John Wiley and Sons, 2006, pp. 375-391.
- [DZC08] R. Dojen, F. Zhang, and T. Coffey, "On the Formal Verification of a Cluster Based Key Management Protocol for Wireless Sensor Networks", *Proceedings of IEEE International on Performance, Computing and Communications Conference*, 2008.
- [EaJ01] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", IFC, 2001. Obtained through Internet: <http://www.faqs.org/rfcs/rfc3174.html> (Accessed 16-07-2012).
- [FAG95] D. D. Falconer, F. Adachi, and B. Gudmundson, "Time division multiple access methods for wireless personal communications", *Communications Magazine*, Vol.33(1), IEEE, 1995, pp.50-57.
- [Fed85] Federal Information Processing Standards, "Computer Data Authentication FIPS-133", NIST, 1985. Obtained through Internet: <http://www.itl.nist.gov/fipspubs/fip113.htm> (Accessed 16-07-2012)
- [Fed01] Federal Information Processing Standards, "Advanced Encryption Standard FIPS-197", NIST, 2001. Obtained through Internet: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf> (Accessed 16-07-2012)
- [Fot10] D. Fotue, "Design of an Enhanced Energy Conserving Routing Protocol Based on Route Diversity in Wireless Sensor Networks", *The 9th IFIP Annual Mediterranean on Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2010
- [GKS03] S. Ganeriwal, R. Kurmar, and M. B. Srivastava, "Timing-sync Protocol for Sensor Networks", *Proceedings of the First International Conference on Embedded Networked Sensor Systems*, 2003, ACM Press, pp. 138-149.

- [GLB03] D. Gay, P. Levis, R. V. Behren, M. Welsh, E. Brewer, and D. Culler, "The Nesc Language: a Holistic Approach to Networked Embedded Systems", Proceedings of the ACM Conference on Programming Language Design and Implementation (PLDI), May, San Diego, CA, 2003, pp.1-11.
- [GRL08] D. Galindo, R. Roman, and J. Lopez, "A Killer Application for Pairings: Authenticated Key Establishment in Underwater Wireless Sensor Networks", Lecture Notes in Computer Science, Vol. 5339, Springer-Verlag, 2008, pp.120-132
- [Gun09] V. C. Gungor, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches", Industrial Electronics, IEEE, Vol. 56(10), 2009, pp. 4258-4265
- [Hat03] L. Hathaway, "National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information", Committee on National Security Systems, June 2003. Obtained through Internet: <http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf> (Accessed 16-07-2012).
- [HCB00] W. B. Heinzelman , A. Chandrakasan, and H. Balakrishnan, "Energy-efficient Communication Protocol for Wireless Microsensor Networks", Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000.
- [HCB02] W. B. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," Wireless Communication, IEEE, Vol. 1(4), Oct. 2002, pp. 660-70.
- [Hea03] S. Heath, |Embedded systems design. EDN series for design engineers (2 ed.)". Newnes. 2003, pp. 11-12. ISBN 9780750655460.
- [HGS11] L. B. Hormann, P. M. Glatz, P.M, C. Steger, and R.Weiss, "Evaluation of Component-Aware Dynamic Voltage Scaling for Mobile Devices and Wireless Sensor Networks", IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011.
- [HKL06] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh, "VigilNet: An

Integrated Sensor Network System For Energy-Efficient Surveillance", *Sensor Networks*, Vol.2(1), ACM, 2006, pp. 1 - 38.

- [HNL07] M. Healy, T. Newe, and E. Lewis, "Efficiently Securing Data on a Wireless Sensor Network," *Journal of Physics: Conference Series*, Vol. 76(1), IOPScience, 2007, pp. 1-6.
- [HPJ02] Y. C. Hu, A. Perrig, and U. B. Johnson, "Wormhole Detection in Wireless Ad Hoc Networks," Department of Computer Science, Rice University, Tech. Rep. TR01-384, 2002.
- [HRH06] A. Hamid, M. O. Rashid, and C. S. Hong, "Routing Security in Sensor Network: HELLO Flood Attack and Defense", *Next-Generation Wireless System*, IEEE, 2006, pp. 77-81
- [Inf02] Information Technology Laboratory, "The Keyed-Hash Message Authentication Code (HMAC)", Federal Information Processing Standards Publication 198, FIP, 2002. Obtained through Internet: <http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf> (Accessed 16-07-2012)
- [Kan01] M. Kanda, "Practical Security Evaluation against Differential and Linear Cryptanalyses for Feistel Ciphers with SPN Round Function", *Lecture Notes in Computer Science*, Vol. 2001, Springer-Verlag, 2001, pp. 324-338.
- [KaS06] J. Kaps and B. Sunar, "Energy Comparison of AES and SHA-1 for Ubiquitous Computing", *Lecture Notes in Computer Science*, Vol. 4097, Springer-Verlag, 2006, pp. 372-381.
- [KaW02] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures", *IEEE International Workshop on Sensor Network Protocols and Applications*, 2002.
- [KGD08] I. Krontiris, T. Giannetsos, and T. Dimitriou, "Launching a Sinkhole Attack in Wireless Sensor Networks: The Intruder Side", *Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 2008.

- [KhS06] E. Khabiri and S. Bettayeb, "Efficient Algorithms for Secure Multicast key Management", Proceedings the 31st IEEE Conference on Local Computer Networks, Nov. 2006, pp. 787-792.
- [KOK09] E. Kohno, T. Ohta, and Y. Kakuda, "Secure Decentralized Data Transfer Against Node Capture Attacks for Wireless Sensor Networks", Proceedings of the International Symposium on Autonomous Decentralized Systems, Athens, Greece, March, 2009. pp. 23-25.
- [KrB04] T. Krag and S. Büettrich, "Wireless Mesh Networking", O'Reilly Wireless Dev Center, O'Reilly, Jan 2004.
- [LDC11a] I. Lasc, R. Dojen, and T. Coffey, "Countering Jamming Attacks Against an Authentication and Key Agreement Protocol for Mobile Satellite Communications", Computers & Electrical Engineering, Vol. 37(2), Elsevier, 2011, pp. 160-168.
- [LDC11b] I. Lasc, R. Dojen, and T. Coffey, "A Mutual Authentication Protocol with Resynchronisation Capability for Mobile Satellite Communications", International Journal of Information Security and Privacy, Vol.5(1), IGI, January 2011, pp. 33-49
- [LDH06] Y. W. Law, J. Doumen, and P. Hartel, "Survey and Benchmark of Block Ciphers for Wireless Sensor Networks," Sensor Networks, Vol. 2(1), ACM, February 2006, pp. 65-93.
- [LDZ10] J. Liu, J. Du, J. Zhang, X. Li, and H. Liu, "RANC: Opportunistic Multi-path Routing Protocol in WSNs Using Reality-Aware Network Coding", Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, 2010, pp.185-190.
- [LHJ05] Y. W. Law, L. V. Hoesel, J. Doumen, P. Hartel, and P. Havinga, "Energy-efficient Link-Layer Jamming Attacks Against Wireless Sensor Network MAC Protocols", Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks, 2005, pp. 76 - 88.

- [LLS09] H. Lee, K. Lee, and Y. Shin, "AES Implementation and Performance Evaluation on 8-bit Microcontrollers", *International Journal of Computer Science and Information Security*, Vol.6(1), IJCSIS, Oct 2009, pp. 70-74.
- [LKS10] J. Lee, K. Kapitanova, and S. H. Son, "The Price of Security in Wireless Sensor Networks", *Computer Networks*, Vol. 5(17), ACM, Dec 2010, pp. 2967-2978
- [LMP05] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An Operating System for Wireless Sensor Networks", *Ambient Intelligence, Part II*, Springer, 2005, pp. 115-148.
- [LNR05] R. Lédeczi, A. Nádas, P. V. Rlyyesi, G. Balogh, B. Kusy, J. Sallai, G. Pap, S. Dóra, K. Molnár, M. Maróti, and G. Simon., "Countersniper System for Urban Warfare", *Sensor Networks*, Vol. 1(2), ACM, 2005, pp. 153-177.
- [LRS02] S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data Gathering Algorithms in Sensor Networks using Energy Metrics," *Parallel and Distrib. Sys.*, Vol. 13(9), IEEE, Sept. 2002, pp. 924-35.
- [Mis10] A. M. Mishra, "Security and QoS in mobile ad hoc wireless networks", Oxford, Wiley-Blackwell, 2010.
- [MKS04] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The Flooding Time Synchronization Protocol", *Proceedings of the International Conference on Embedded Networked Sensor Systems*, 2004.
- [MMB05] S. D. Muruganathan, D. C. F Ma, R. I. Bhasin, and A. O. Fapojuwo, "A centralized energy-efficient routing protocol for wireless sensor networks", *Communications Magazine*, Vol.43(3), IEEE, March 2005, pp. 8-13.
- [Moh04] R. Mohamad, "Network security architecture", *Journal of Computing Sciences in Colleges*, Vol. 19(4), CSCC, 2004, pp. 307-313.
- [MOJ06] A. Milenković, C. Otto, and E. Jovanov, "Wireless Sensor Networks for Personal Health Monitoring: Issues and an Implementation", *Computer Communications*, Vol.29(13-14), Elsevier, 2006, pp.2521-2533.

- [MRK05] V. P. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff, "A Minimum Cost Heterogeneous Sensor Network with a Lifetime Constraint", *Mobile Computing*, Vol.4(1), IEEE,2005, pp. 4-15
- [Nat77] National Bureau of Standards, "Data Encryption Standard", FIPS-Pub.46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.
- [Niy09] Niyazpk, "Advanced Encryption Standard (AES) Implementation in C/C++ with comments". 2009. Obtained through Internet: <http://www.hoozi.com/post/829n1/advanced-encryption-standard-aes-implementation-in-c-c-with-comments-part-1-encryption> (Accessed 16-07-2012)
- [NLD08] P. Ning, A. Liu, and W. Du, "Mitigating DoS Attacks Against Broadcast Authentication in Wireless Sensor Networks", *Sensor Networks*, Vol. 4(1), Article 1, ACM, January 2008, 34 pages.
- [NLL06] E. C. H. Ngai, J. Liu, and M. R. Lyu, "On the Intruder Detection for Sinkhole Attack in Wireless Sensor Networks", *IEEE International Conference on Communications*, June 2006.
- [OFV07] L. B. Oliveira, A. Ferreira, M. A. Vila, H. C. Wong, M. Bern, R. D. and A. A. F. Loureiro, "SecLEACH-On the Security of Clustered Sensor Networks", *Signal Processing*, Vol. 87(12), Elsevier, December 2007, pp. 2882-2895.
- [Oth08] M. Othman, "Principles of Mobile Computing and Communications", Auerbach Publications, ISBN-10: 1420061585, 2008.
- [PaL05] K. Pahlavan and A. H. Levesque, "Wireless Information Networks - 2nd Edition", Wiley - Interscience, ISBN: 0-471-72542-0, 2005
- [PDC08] V. Pasca, D. Dojen, T. Coffey, and R. Gyorodi, "Detecting Attacks on Security Protocols Using Model Checking", 7th International Conference on Renewable Sources and Environmental Electro-Technologies, Oradea, Romania, 29-30 May 2008, pp. 71-76

- [PHC06] A.S.K Pathan, L. Hyung-Woo, and S.H. Choong, "Security in Wireless Sensor Networks: Issues and Challenges", Proceedings of the 8th International Conference on Advanced Communication Technology, Feb 2006.
- [PLP06] K. Piotrowski, P. Langendoerfer, and S. Peter, "How Public Key Cryptography Influences Wireless Sensor Node Lifetime", Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks, 2006.
- [PrS11] N. Pradhan and T. Saadawi, "Energy Efficient Distributed Power Management Algorithm with Directional Antenna for Wireless Sensor Networks", 34th IEEE Sarnoff Symposium, 2011
- [PSW04] A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks", Wireless Sensor Networks, Vol. 47(6), ACM, 2004, pp. 53-57.
- [Riv92] R. L. Rivest, "The MD5 Message-Digest Algorithm", RFC-1321, MIT Laboratory for Computer Science and RSA Data Security, Inc, 1992.
- [Riv94] R. L. Rivest, "The RC5 Encryption Algorithm", Proceedings of the Second International Workshop on Fast Software Encryption (FSE) 1994. pp. 86-96.
- [RMB09] D. R. Raymond, R. C. Marchany, M. I. Brownfield, and S. F. Midkiff, "Effects of Denial-of-Sleep Attacks on Wireless Sensor Network MAC Protocols", Vehicular Technology, Vol. 58(1), IEEE, 2009, pp. 367-380.
- [RMG02] P. Rentala, R. Musunuri, S. Gandham, U. Saxena, "Survey on Sensor Networks", Technical Report, UTDCS-33-02, University of Texas at Dallas, 2002.
- [Rog04] P. Rogaway, "Nonce-based symmetric encryption", In: Roy, B., Meier, W. (eds.) LNCS, vol. 3017, Springer, Heidelberg, 2004, pp. 348-359.
- [RoM04] K. Romer and F. Mattern, "The Design Space of Wireless Sensor Networks", Wireless Communications, Vol.6(11), IEEE, Dec 2004, pp. 54-61.
- [SaK09] N. Satoh and H. Kumamoto, "Analysis of Information Security Problem by Probabilistic Risk Assessment", International Journal of Computer, Vol.3(3), CISJ, 2009, pp. 337-347.

- [SEF10] P. Soochang, L. Euisin, Y. Fucai, K. Sang-Ha, "Scalable and Robust Data Dissemination for Large-Scale Wireless Sensor Networks", *Consumer Electronics*, Vol.56(3), IEEE, 2010, pp. 1616-1624.
- [SFQ08] L. Shen; H. Feng, Y. Qiu, and H. Ding, "A New Kind of Cluster-based Key Management Protocol in Wireless Sensor Network", *Proceedings of the IEEE Conference of Networking, Sensing and Control*, 2008.
- [ShP04] E. Shi and A. Perrig, "Designing Secure Sensor Networks", *Wireless Communications*, Vol.11(6), IEEE, 2004, pp. 38-43.
- [SiC01] A. Sinha and A. Chandrakasan , "Dynamic Power Management in Wireless Sensor Network", *Design and Test of Computer*, Vol.18(2), IEEE, 2001, pp. 62-74.
- [SJS10] V. P. Singh, S. Jain, and J. Singhai, "Hello Flood Attack and its Countermeasures in Wireless Sensor Networks", *International Journal of Computer Science Issues (IJCSI)*, Vol. 7(11), May 2010. pp. 23-27.
- [SKS11] "T. Singh, V. Kumar, K. Saxena, and A. Saxena, ""Evaluation of Security Conditions of Protocols for Data Routing in Wireless Sensors Networks"", *International Journal of Soft Computing and Engineering (IJSCE)*, Vol. 1(1), 2011, pp. 33-42.
- [Slo05] E. Sloan, "Security and Defence in the Terrorist Era", *McGill-Queen's University Press*, Montreal, 2005.
- [SML03] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for Internet applications", *Networking*, Vol.11(1), IEEE, 2003, pp. 17-32.
- [SMP04] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An Analysis of a Large Scale Habitat Monitoring Application," *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems.*, November 2004
- [SRS07] R. Srinath, R. A.V. Reddy, and R. Srinivasan, "AC: Cluster Based Secure Routing Protocol for WSN", *Proceedings of International Conference on Networking and Services*, 2007.

- [Sta07] W. Stallings, "Cryptography and Network Security: Principles and Practice (5th Edition)", Prentice Hall, ISBN. 0136097049, 2010.
- [Ste07] M. M. J. Stevens, "On Collisions for MD5", 2007. Obtained through Internet: <http://www.win.tue.nl/hashclash/On%20Collisions%20for%20MD5%20-%20M.M.J.%20Stevens.pdf> (Accessed 16-07-2012)
- [Ste97] B. Sterzbach, "GPS-based Clock Synchronization in a Mobile, Distributed Real-Time System", *Journal of Real-Time Systems*, Vol. 12(1), Springer, 1997, pp. 63-75.
- [SWC02] R. Szewczyk, V. Wen, D. Culler, and D. Tygar, "SPINS: Security Protocols for Sensor Networks", *Journal of Wireless Networks (WINE)*, Vol. 8(5), Kluwer Academic Publishers Hingham, 2002, pp 521-534.
- [SWC07] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter, "LUSTER: Wireless Sensor Network for Environmental Research", *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*. November 2007.
- [TAH02] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A Taxonomy of Wireless Micro-Sensor Network Models", *SIGMOBILE Mobile Computing and Communications Review*, Vol.6(2), ACM, April 2002, pp. 28 - 36.
- [Tex07] Texas Instrument, "CC2520 Datasheet", 2007. Obtained through Internet: <http://www.ti.com/lit/ds/symlink/cc2520.pdf> (Accessed 16-07-2012)
- [Tuc97] W. Tuchman, "A brief history of the data encryption standard". *Internet besieged: countering cyberspace scofflaws*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA. 1997, pp. 275-280
- [Uou02] I. R. Uouceur, "The Sybil Attack", *International Workshop on Peer-to-Peer Syslcmr*, 2002.
- [Vac09] J. R. Vacca, "Computer and Information Security Handbook", ISBN-10: 0123743540, Morgan Kaufmann, 2009

- [ViP07] A. Vitaletti and G. Palombizio, "Rijndael for Sensor Networks: is Speed the Main Issue?" *Electronic Notes in Theoretical Computer Science*, Vol. 171(1), Elsevier, April 2007, pp. 71-81.
- [Wan08] D. Wang, "An energy-efficient cluster head assignment scheme for hierarchical wire-less sensor networks", *Int. Journal of Wireless Inf. Networks*, Vol.15(2), Springer, 2008 pp.61-71
- [WAR06] Y. Wang, G. Attebury, and B. Ramamurthy, "A Survey of Security Issues in Wireless Sensor Networks", *Communications Surveys*, Vol. 8(2), IEEE, 2006, pp. 2-23.
- [WLS06] J.P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless Sensor Networks Security: A Survey", In *Security in Distributed, Grid, and Pervasive Computing*, CRC Press, 2006, pp. 1-50.
- [XCS06] Y. Xiao, H. Chen, B. Sun, R. Wang, S. Sethi, "MAC Security and Security Overhead Analysis in the IEEE 802.15.4 Wireless Sensor Networks", *EURASIP Journal on Wireless Communications and Networking*, Vol. 2006(2), Springer, April 2006, pp. 1-12
- [XRC04] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network for Structural Monitoring", *Proceedings of the 2nd international conference on Embedded networked sensor systems*. 2004.
- [XWD09] X. Xiong, D. S. Wong, and X. Deng, "TinyPairing: Computing Pairing on Sensor Nodes with Higher Speed and Less Memory", *Proceedings of the IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, July 2009.
- [YBH10] Y. Yang, E. Bai, J. Hu, and W. Wu, "MRBCH: A Multi-Path Routing Protocol Based on Credible Cluster Heads for Wireless Sensor Networks", *International Journal of Communications, Network and System Sciences*, Vol. 3(8), Scientific Research, 2010. pp. 689-696.
- [You04] O. Younis, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks", *Mobile Computing*, Vol.3 (4), IEEE, 2004, pp.366-379

- [ZDC11] F. Zhang, R. Dojen, and T. Coffey, "Comparative performance and energy Consumption Analysis of Different AES Implementations on a Wireless Sensor Network Node". Accepted for Publication in the International Journal of Sensor Networks (IJSNET), Inderscience (accepted 20.06.2011)
- [Zha10] J. Zhang, "Multi-path Routing Protocol of Node Cover Layer in WSN", Computer Engineering and Applications. Vol. 46(8), Mar. 2010. pp. 87-91.
- [ZhN08] F. Zhang and Y. Niu, "Rijndael Arithmetic Analyse and Optimize", Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, Dalian, China, October 2008.