EMBEDDED BASED CRYPTOGRAPHIC MODULE FOR LOW POWER

WIRELESS SENSOR NODES COMPLYING WITH FIPS 140-2

A Thesis in
Electrical Engineering
by
Jesal H. Kanani

© 2009 Jesal H. Kanani

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2009
The thesis of Jesal H. Kanani was reviewed and approved* by the following:

Sven G. Bilén
Associate Professor of Engineering Design, Electrical Engineering, and Aerospace Engineering
Thesis Advisor

Julio Urbina
Assistant Professor of Electrical Engineering

Kenneth Jenkins
Head of Department
Professor of Electrical Engineering

*Signatures are on file in the Graduate School.
Abstract

The work presents a framework for a Cryptographic Module for wireless sensor nodes complying with FIPS 140-2, the Federal Information Processing Standard (FIPS), which requires cryptographic-based security in systems that deal with sensitive and valuable data. The standard provides various requirements covering areas related to secure design and implementation of a Cryptographic Module. As this module will be installed on Navy ships, FIPS validation is required. The module is being implemented on a Texas Instrument MSP430 microprocessor, and the ciphering algorithm used is the Advanced Encryption Standard (AES) as it is a FIPS-approved algorithm. The module will be primarily responsible for the input/output of binary data to and from the Master Controller and ciphering and deciphering of this data using the AES ciphering algorithm along with various authentication and error checking techniques. The presented architecture and, hence, the Cryptographic Module can be used for several other applications in addition to use in sensor nodes.
Table of Contents

List of Figures .................................................................................................................. vii
List of Tables .................................................................................................................. viii
Nomenclature .................................................................................................................. ix
Acknowledgments .......................................................................................................... xi

Chapter 1 Introduction ................................................................................................... 1

1.1 Cryptographic Module – Evolution ......................................................................... 1
1.2 Cryptanalysis and Establishment of FIPS ............................................................... 3
1.3 OSI/TCP Model and Data Link Layer .................................................................... 4
1.4 Wireless Sensor Nodes .......................................................................................... 5
1.6 System Outline ...................................................................................................... 6
1.7 Contribution ........................................................................................................... 7
1.8 Thesis Outline ........................................................................................................ 8

Chapter 2 Background .................................................................................................. 9

2.1 Literature Survey .................................................................................................. 9
2.1.1 Selection of Microprocessor for Power Efficiency ............................................ 10
2.1.2 Power Consumption with Respect to Ciphering Algorithms ......................... 11
2.1.2.3 Power Consumption Based on Size of Packet ........................................... 12
2.1.2.4 Power Consumption Based on Size of Key ............................................... 13
2.1.2.4 Power Consumption Based on Number of Cipher Rounds. ....................... 14
2.2 Federal Information Processing Standard (FIPS) ................................................... 15
2.2.1 Security Levels for FIPS 140–2 Certification .................................................... 16
2.2.1.1 Security Level 1 ....................................................................................... 16
2.2.1.2 Security Level 2 ...................................................................................... 16
2.2.1.3 Security Level 3 ...................................................................................... 17
2.2.1.4 Security Level 4 ...................................................................................... 17
2.2.2 Requirements for Certification for Various Levels ........................................... 19
2.2.2.1 Cryptographic Module Specification ....................................................... 19
2.2.2.2 Cryptographic Module (CM) Ports and Interfaces .................................. 20
2.2.2.3 Roles, Services, and Authentication ....................................................... 20
List of Figures

Figure 1-1: Basic block diagram of a Cryptographic Module ........................................... 3
Figure 1-2: The 7-layer OSI model (Tanenbaum, 2002) .................................................. 4
Figure 1-3: Typical architecture of a sensor node. (Upload Wikipedia) ............................... 6
Figure 1-4: Sensor node mesh network .............................................................................. 7
Figure 2-1: Energy Consumption for different ciphering algorithms as a function of packet size (Kiratiwintakorn, 2005) ..................................................................................... 12
Figure 2-2: Energy Consumption for different ciphering algorithms as a function of key size (Kiratiwintakorn, 2005) ..................................................................................... 13
Figure 2-3: Energy Consumption for different ciphering algorithms as a function of key size (Kiratiwintakorn, 2005) ..................................................................................... 14
Figure 2-4: Time consumption of Encryption algorithm (Abdul Mina, Abdual Kader, & Hadhoud, 2009) .................................................................................................................. 15
Figure 2-5: States in a Cryptographic Module ....................................................................... 23
Figure 3-1: System Overview .............................................................................................. 27
Figure 3-2: Detailed system architecture depicting the four modules .................................. 28
Figure 3-3: End-to-end system between sensor nodes .......................................................... 32
Figure 3-4: System interface overview .................................................................................. 33
Figure 3-5: Interface between Cryptographic Module and Master MSP430 chip ............... 33
Figure 3-6: Pin interface between the Master and Crypto MSP430F1611s ......................... 34
Figure 4-1: Ciphering process using AES algorithm ............................................................ 40
Figure 4-2: Diffie–Hellman key exchange process .............................................................. 42
Figure 4-3: Algorithm for zeroization of RAM ................................................................. 43
Figure 4-4: Algorithm for Known Answer Tests (KATs) .................................................... 44
Figure 4-5: Flowchart depicting the flow of data packets in the system .............................. 46
Figure 5-1: MSP-TS430PM64 target board by Texas Instruments (Texas Instruments) .... 55
Figure 5-2: Experimental set-up for emulating Master-Crypto function .............................. 56
Figure 5-3: Experimental setup ............................................................................................ 57
Figure 5-4: Screenshot of the IAR with print statements ..................................................... 58
Figure 8-1: Pseudo Code for AES (NIST AES) ................................................................. 72
Figure 8-2: SubBytes( ) function in AES (NIST AES) ..................................................... 72
Figure 8-3: ShiftRows( ) function of the AES (NIST AES) ............................................... 72
Figure 8-4: MixCoulmns( ) function (NIST AES) of the AES .......................................... 72
Figure 8-5: AddRoundKey( ) function (NIST AES) of the AES ...................................... 72
List of Tables

Table 2-2: Cryptographic Module requirement stringency vs. levels of certification ..... 18
Table 2-3: Summary of physical security requirements (NIST, 2009) .......................... 24
Table 3-1: Details of the sensor subsystem .................................................................. 29
Table 3-2: Details of the Master MSP430 chip. .......................................................... 30
Table 3-3: Details of the Cryptographic Module ......................................................... 31
Table 3-4: Details of the Dust Radio .......................................................................... 31
Table 4-1: Identifier bits and the corresponding payload in the packet ...................... 45
Table 7-1: Heartbeat Message elements .................................................................... 69
Table 7-2: Configuration Message elements ............................................................... 69
Table 7-3: Firmware Status Message Elements ......................................................... 69
Table 7-4: Waveform Body Message ......................................................................... 70
Table 8-1: Key-Block-Round Combinations (NIST AES) .............................................. 71
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analog-to-Digital Converter</td>
</tr>
<tr>
<td>AES</td>
<td>Advanced Encryption Standard</td>
</tr>
<tr>
<td>AVG</td>
<td>Average</td>
</tr>
<tr>
<td>CSP</td>
<td>Critical Security Parameters</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial Off The Shelf</td>
</tr>
<tr>
<td>CLK</td>
<td>Clock</td>
</tr>
<tr>
<td>CISC</td>
<td>Complex Instruction Set Computing</td>
</tr>
<tr>
<td>CMS</td>
<td>Condition Monitoring System</td>
</tr>
<tr>
<td>CM</td>
<td>Cryptographic Module</td>
</tr>
<tr>
<td>CO</td>
<td>Crypto Officer</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>EAL</td>
<td>Evaluation Assurance Level</td>
</tr>
<tr>
<td>EMC</td>
<td>Electro-Magnetic Compatibility</td>
</tr>
<tr>
<td>EMI</td>
<td>Electro-Magnetic Interference</td>
</tr>
<tr>
<td>FIPS</td>
<td>Federal Information Processing Standards</td>
</tr>
<tr>
<td>FCS</td>
<td>Frame Check Sum</td>
</tr>
<tr>
<td>IBO</td>
<td>Identity-Based Operator</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electronic and Electrical Engineers</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
<tr>
<td>KAT</td>
<td>Known Answer Test</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>NA</td>
<td>Not Available</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>OSI</td>
<td>Open System Interconnect</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PUB</td>
<td>Publication</td>
</tr>
<tr>
<td>PIN</td>
<td>Personal Identification Number</td>
</tr>
<tr>
<td>PIC</td>
<td>Programmable Interface Controller</td>
</tr>
<tr>
<td>RSA</td>
<td>Rivest, Shamir, and Adleman ciphering algorithm</td>
</tr>
<tr>
<td>RNG</td>
<td>Random Number Generator</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computing</td>
</tr>
<tr>
<td>RC4</td>
<td>Rivest Cipher 4</td>
</tr>
<tr>
<td>RC5</td>
<td>Rivest Cipher 5</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>TI</td>
<td>Texas Instruments</td>
</tr>
<tr>
<td>UID</td>
<td>Unique Identifier</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver/Transmitter</td>
</tr>
</tbody>
</table>
USB  Universal Serial Bus  
ZIF  Zero Insertion Force  

Symbols

µ  Micro  
@  At the Rate  
+  Addition  
++  Increment  
*  Pointer  
∥  Logical OR  
<  Less Than Operator  
#  Number  

x
Acknowledgments

I am most thankful and indebted to my advisor, Dr. Sven Bilén for providing me with an opportunity to learn and work under him during my stint as a graduate student. He has always been a source of inspiration, has shown utmost commitment to me and my work, and provided me with all the help he possibly could. I take this opportunity to thank him for believing in me and helping me when I needed it the most.

I am also grateful to Prof. Julio Urbina and Bob Capuro for providing me with their feedback and timely suggestions. I thank the Graduate office staff MaryAnn, Gabi, and SherryDawn for their help with the administrative paperwork and always assisting me with my deadlines. I would like to express my gratitude to my fellow colleague Ravikant Gupta, who assisted me in carrying out the experiments. His timely inputs and suggestions have helped me enormously throughout the duration of this research. Additionally, I would like to thank Jacob and Jeremy at KCF Technologies for their suggestions and support.

Lastly, to my wonderful family, I want to express my deepest love and gratitude. My parents, dadi, and my sister have always supported me through thick and thin and deserve all the credit for what I am today. Thank you, Niki, for being a wonderful girlfriend and for supporting and understanding me all these years while I was busy with my research at Penn State.
Chapter 1

Introduction

This chapter gives an introduction to the concept of cryptography and modern cryptographic modules. It also gives a brief overview of sensor nodes used in wireless sensor networks. The chapter gives an introduction to FIPS and the reason for its development. FIPS has various requirements and some of them will be described herein.

1.1 Cryptographic Module – Evolution

Ciphering is one of the most popular and increasingly implemented methods for achieving security in communication systems. The need for sending messages between two parties without worrying about a third party interpreting the message has driven the evolution of ciphering algorithms. From the simple Caesarean shift algorithm to the Enigma machine, through RSA encryption to quantum cryptography, the domain has made substantial leaps toward the ultimate goal of an absolutely indecipherable encryption scheme.

Cryptography is the science of scrambling data in order to prevent unauthorized parties from deciphering and reading the content of that data. Cryptography can be divided into two categories: transposition and substitution (Tanenbaum, 2002). Transposition involves the systemic swapping of information within a data set. As cryptography has evolved, the methods have become more and more advanced to counter cryptanalysis techniques.
As more and more data is digitized and stored in modern computers, such data is usually exchanged in the digital form. Based on the sensitivity of this data, the need to protect the data from theft increases. Data transmission on wireless channels is convenient at the same time it requires being protected from falling into wrong hands. All these requirements lead to a requirement of developing efficient ways of protecting the data. One of the most obvious ways is to cipher this data.

Ciphering has become increasingly computer oriented and modern day cryptographic algorithms require significant computational resources. These cryptographic algorithms are implemented on processors with the hardware and software capabilities for executing the computations. The result is the modern day Cryptographic Module—segments of hardware or software programs or an integration of the two—which essentially receives plain text data, ciphers it using the implemented cryptographic algorithms, and then outputs the ciphered data. Thus, the Cryptographic Module essentially can be viewed as a black box that takes in data, processes it, and outputs a ciphered version of the input data. The reverse process is also true, i.e., the black box (Cryptographic Module) takes in the ciphered data, works on it, and outputs the original plain text.

As mentioned earlier, these Cryptographic Modules must be secure such that the ciphered data can be ideally deciphered only by the module that ciphered it. This leads to a series of requirements that, after much research, led to the evolution of the Federal Information Processing Standards (FIPS), as discussed in the following sections.
1.2 Cryptanalysis and Establishment of FIPS

The need for a secure ciphering module that essentially provides a secure environment for implementing cryptographic algorithms was established in the preceding section. Cryptanalysis is the science of breaking cryptography. The methods and techniques of cryptanalysis have changed drastically along with the development of cryptography, adapting to increasing cryptographic complexity. The results of cryptanalysis have also changed—it is no longer possible to have unlimited success in code breaking, and there is a classification of what constitutes a rare practical attack. The Figure 1-1 shows a block diagram of a typical Cryptographic Module.

As the need of ciphering is ever increasing in the military and other agencies, it is essential to employ standards for flexibility, security strength, development environment, module interfaces, etc. Accordingly, the National Institute of Standards and Technology (NIST) released the Federal Information Processing Standard (FIPS) in January 1994 (NIST, 2009). The standard specifies the security requirements that must be satisfied by a Cryptographic Module within a security system protecting sensitive data.
1.3 OSI/TCP Model and Data Link Layer

The Open System Interconnection (OSI) model is used to explain communication of data between networked systems. In this model, the entire communication process takes place in 7 layers (or less). Of the 7 layers, 3 layers concentrate on the actual transfer of data from between systems, while 4 layers are concerned with individual systems to complete the data transfer at their respective ends. The primary benefits of using the OSI model is that it helps the user trace an end-to-end picture of the entire data communication, increases ease of troubleshooting, and is easier to integrate newer technologies as they are developed (Simoneau, 2006). The Figure 1-2 shows the seven layers of the OSI model.

![Figure 1-2: The 7-layer OSI model (Tanenbaum, 2002)](image)

We are particularly interested in the 2nd layer, which is called the ‘Data Link Layer’. FIPS 140-2 requires the Cryptographic Module to implement a Layer 2 encryption. Thus,
we will focus our discussion on the Data Link Layer. Layer 2 of the OSI model executes the following functions (Simoneau, 2006):

- allowing a device to access the network it is attached to, for any kind of data transfers;
- providing addressing to the device so that the data can be routed in the network; and
- implementing error detection algorithms.

1.4 Wireless Sensor Nodes

A sensor node is also sometimes called a “mote”, a term coined by researchers at Berkeley NEST and CENS. A mote is a node in a wireless sensor network that usually generates some data using the sensors, sometimes processes this data (as in our case), and then communicates it to other motes in the network. A typical architecture of a sensor node is depicted in Figure 1-3. As shown in the Figure 1-3, a mote typically consists of a microprocessor, a sensor device, memory, power source, and other modules also depending on the functionality of the node. The S5NAP™ sensor node, a proprietary design by Impact RLW Inc., was used in this work. Additional information about the S5NAP nodes is presented Chapters 3 and 4.
1.6 System Outline

The Cryptographic Module that is developed in this work will work in collaboration with other modules in the system and make the entire end-to-end system secure for data communication. Such a Cryptographic Module can be integrated into existing mesh network of sensor nodes (KCF Technologies, 2008) as depicted in Figure 1-4. The sensor nodes are points that collect data ranging from temperature, vibration, humidity, etc. These data must be stored or transmitted to other nodes. The sensor node is self power generating with the help of the vibration-based energy harvesting technology.

The data generated by the node is passed to the on-chip processor, which essentially ciphers the data and sends it out to other nodes in the network using a radio link. The on-chip processor consists of a pair of two Texas Instruments MSP430 microprocessors, hereafter referred to as the MSP430 crypto-module, which act in the Master–slave
configuration, one obtaining the data from the nodes, while the other is solely responsible for ciphering of the data. We will call this the MSP430 crypto-module.

1.7 Contribution

The project aims at developing a Cryptographic Module that can be used as a black box responsible for encoding sensitive data. Our contribution through this project has been in developing software for NIST recommended algorithms required for FIPS 140-2 certifications. The software developed has been designed to operate in low power modes for the MSP430 processor thus making the codes power efficient. The codes are optimized for reduced usage of instruction cycles for minimizing power consumption.

The system developed can be implemented on any wireless sensor network irrespective of the routing protocols used. There is enough room in the system for future
expansion or implementing parallel communication protocols on the module. Our contribution has been in designing such a low power module that can be used not just for wireless sensor networks, but for any portable device that requires data protection and requires low power consumption.

1.8 Thesis Outline

The remainder of this thesis is divided into a total of five chapters. Each chapter covers a broad range of topics and provides an insight on the project with relevant tables and pictures.

Chapter 2 covers the literature review done for the project and gives a detailed description of FIPS and its requirements for a Cryptographic Module.

Chapter 3 includes the system architecture and gives a detailed description of various modules in the system and their respective roles. It also describes the numerous interfaces in the system between the modules described in the architecture.

Chapter 4 covers the implementation of the algorithms required for the FIPS certification of the Cryptographic Module. It also covers in detail the project-specific system developed and its proposed implementation on the MSP430 processors.

Chapter 5 covers the testing done on the hardware. It contains various screenshots explaining the implementation of the software on the development boards used design the Cryptographic module.

Chapter 6 provides a conclusion and suggests some future work that can be implemented on the Cryptographic Module to make it a viable commercial product.
Chapter 2

Background

This chapter is divided into two parts. The first part gives a summary of relevant work found in the literature on the work done by others on similar modules and other cryptographic devices. The chapter sheds light on various ways of implementing different algorithms used in the project and its impact on the system. The chapter also includes the literature necessary for understanding the power optimization in the codes required on the MSP430 and various papers and technical documents on the TI products.

NIST specifies the security requirements that shall be satisfied by Cryptographic Modules intended for use by federal agencies. The second part of this chapter gives a detailed description of one such standard issued by the NIST called the Federal Information Processing Standard (FIPS). The chapter gives the background that led to the development of FIPS and the various requirements associated with FIPS certification. Finally the chapter sheds light on the various levels of certifications available and the strictness of the compliance of requirements pertaining to that level of certification.

2.1 Literature Survey

The aim of the project is to design a cryptographic module that can implement certain algorithms required by NIST on all incoming data. Besides restrictions on the types of algorithm implemented, the cryptographic module is also required to have power efficiency and has memory constraints.
2.1.1 Selection of Microprocessor for Power Efficiency

After decades of development, the microprocessor remains a widely used central brain of hardware systems. Two major microprocessor architectures have made this possible; Complex Instruction Set Computing (CISC) and the more cost-effective Reduced Instruction Set Computing (RISC) (Patterson & Ditzel, 1980).

The primary method of decreasing power consumption by a processor while executing an instruction will be by reducing the number of clock cycles required to execute an instruction (Piguet, Schneider, Masgonty, Arm, Durand, & Stegers, 2004). Before selecting the microprocessor that was used in this project, we performed a comparative study of the available processors in the market. The PIC series of microprocessors by Microchip are quite power efficient, in particular the PIC16F87 processor, which is an 8-bit processor (Microchip, 2009). The low power features include 4 power managed modes and two-speed oscillators. Similarly, we reviewed the power consumptions expected for many leading commercially available microprocessors. The details as per the data sheet provided by the respective companies are produced in Table 2-1. As seen in this table, based on the power consumption figures available, we selected the Texas Instrument MSP430Fxxx series microprocessor due to its best overall power consumption pattern.

<table>
<thead>
<tr>
<th>Company</th>
<th>Processor</th>
<th>8 or 16 bit</th>
<th># of Power Modes</th>
<th>Power Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mode 1</td>
</tr>
<tr>
<td>Microchip</td>
<td>PIC16F87/88</td>
<td>8 bit</td>
<td>4</td>
<td>152 μW @ 1 MHz</td>
</tr>
<tr>
<td>Atmel</td>
<td>AT89C5115</td>
<td>8 bit</td>
<td>3</td>
<td>3.7 mW @ 1 MHz</td>
</tr>
<tr>
<td>Texas Instruments</td>
<td>MSP430F1611</td>
<td>16 bit</td>
<td>3</td>
<td>726 μW @ 1 MHz</td>
</tr>
<tr>
<td>Maxim</td>
<td>MAXQ2000</td>
<td>16 bit</td>
<td>2</td>
<td>4.75 mW @ 14 MHz</td>
</tr>
<tr>
<td>EM Microelectronic</td>
<td>EM6812</td>
<td>8 bit</td>
<td>4</td>
<td>360 μW @ 1 MHz</td>
</tr>
</tbody>
</table>

2.1.2 Power Consumption with Respect to Ciphering Algorithms

The strength of a cryptographic algorithm can only be stated if it has been rigorously tested by cryptographers over a long period of time. RC4 is a strong as far as computation is concerned, but it has its drawbacks (Grosul & Wallach, 2000). Other strong algorithms are RC5, Blowfish, DES, and most recently AES.

Energy consumption by a cryptographic algorithm is dependent on a number of parameters. Some of these are the number of cycles used to perform the ciphering, size of the key used, or the size of the block of data ciphered. In the case where a packet is divided into a number of smaller packets, these smaller packets are ciphered individually.
The following sections describe the effect of these three parameters on the power consumption of different popular ciphering algorithms.

### 2.1.2.3 Power Consumption Based on Size of Packet

There is a trade-off between transmission efficiency of small packets over large packets. However, large packets have an increased chance of developing errors during transmission as compared to smaller packets (Lettieri, Schurgers, & Srivastava, 1999). However, the ciphering of larger packets consumes less energy as compared to smaller packets (Prasithsangree & Krishnamurthy, 2003). Figure 2-1 shows the power consumption of cryptography for different size packets. Here the criteria of key length and rounds of ciphering have been kept constant for a single type of ciphering.

![Figure 2-1: Energy Consumption for different ciphering algorithms as a function of packet size (Kiratiwintakorn, 2005)](image)
2.1.2.4 Power Consumption Based on Size of Key

Based on the type of algorithm, the key is employed in different ways. For cryptographic algorithms like AES, the larger the key, the larger the number of rounds to be performed for ciphering. This is why, as the size of the key increases, the energy consumed also increases to cipher the same size of packet (Standaert, Berna, & Preneel, 2004). Conversely, with algorithms like RC4 and RC5, the energy consumption is independent of the size of the key. This is because both the RC4 and the RC5 encryption are dependent on a RNG and production of a key from an RNG does not consume too much power (Vocal Technologies, 2003). As can be seen from the Figure 2-2, the above theory holds true.

Figure 2-2: Energy Consumption for different ciphering algorithms as a function of key size (Kiratiwintakorn, 2005)
2.1.2.4 Power Consumption Based on Number of Cipher Rounds.

Block ciphers like AES or RC5 perform similar ciphering rounds a number of times based on the size of the key or size of block. Each round requires execution of instructions, and so it is obvious that as the number of rounds increases, the energy consumed by the ciphering algorithm increases. Thus, to save power we reduce the number of rounds performed. However, this adversely affects the strength of the cryptography and it is proved that the number of rounds has a direct impact on attacks like linear cryptoanalysis (Matsui, 1994) and differential cryptoanalysis (Biham & Shamir, 1991). In Figure 2-4 we can see the change in the power consumption as a function of the number of rounds performed in ciphering.

![Figure 2-3: Energy Consumption for different ciphering algorithms as a function of key size (Kiratiwintakorn, 2005)](image-url)
An important factor to look at would be the time required for ciphering the packet. There is no significant difference in time in base 64 and hexadecimal encoding (Abdul Mina, Abdual Kader, & Hadhoud, 2009). Figure 2-4 shows the time consumption of encryption algorithms in base 64 encoding.

![Figure 2-4: Time consumption of Encryption algorithm](image)

2.2 Federal Information Processing Standard (FIPS)

The cryptographic modules that conform to these specified rules mentioned by NIST are awarded a FIPS certification, the most current being Version 140-2 (National Institute of Standard and Technology, 2002). The certification level awarded depends on which security conditions are satisfied, with higher levels needing to meet stricter requirements.
2.2.1 Security Levels for FIPS 140–2 Certification

There are four different levels of certification in FIPS certification, with Level 1 being the easiest to achieve up to Level 4, which is the strictest. Although different security elements may achieve higher levels of FIPS certification, the entire module will be certified at the security level of the lowest element’s level.

2.2.1.1 Security Level 1

Security Level 1 provides the lowest level of security, primarily due to the fact that there are no physical security requirements. An example of a Security Level 1 Cryptographic Module could be a Personal Computer Encryption Board (NIST, 2009). Security Level 1 may be adequate for low-level security applications in combination with other security arrangements, or in a closely guarded environment where the probability of a security breach is relatively low. Implementation of this level of security is more cost effective and thus can be implemented on general-purpose large-scale modules.

2.2.1.2 Security Level 2

Security Level 2 provides better physical security than that of Level 1. Here, tamper evidence for the module is required, which could be in the form of coatings, seals, or pick-resistant locks (NIST, 2009). Also, there are role-based users’ requirements at this level. Overall, the module needs to implement stricter algorithms, physical security arrangements, and access protocols for Security Level 2 as compared to a Level 1.
2.2.1.3 Security Level 3

Security Level 3 has requirements for additional “easy to detect and respond to” types of tamper-evident safeguards. In addition, Level 3 requires mechanisms for preventing access to Critical Security Parameters (CSPs) by an intruder. Overall, the certification requirements for Security Level 3 are much greater than those of Levels 1 and 2, which makes achieving Level 3 certification less cost-effective than achieving Levels 1 or 2. However, this level of certification is required when the Cryptographic Module is going to be installed in a very hostile environment or the module will be processing extremely sensitive data.

2.2.1.4 Security Level 4

Security Level 4 provides the highest level of security defined by the FIPS standard. Level 4 requires complete protection around the Cryptographic Module (National Institute of Standard and Technology, 2002) (NIST, 2009) with preventive and detection capabilities for any possible security breach. At this level, a security module is also required to comply with environmental failure testing and security of CSPs in an advent of a security breach. This security level is required for the most sensitive type of data processing and when the module is placed in a very hostile environment.

Table 2-2 summarizes the different requirements for various levels of certification in FIPS140-2.
Table 2-2: Cryptographic Module requirement stringency vs. levels of certification

| Table 2-2: Cryptographic Module requirement stringency vs. levels of certification |
|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| **Security Level 1**             | **Security Level 2**             | **Security Level 3**             | **Security Level 4**             |
| **Cryptographic Module Specification** | Specification of cryptographic module, cryptographic boundary, Approved algorithms, and Approved modes of operation. Description of cryptographic module, including all hardware, software, and firmware components. Statement of module security policy. | | |
| **Cryptographic Module Ports and Interfaces** | Required and optional interfaces. Specification of all interfaces and of all input and output data paths. | Data ports for unprotected critical security parameters logically or physically separated from other data ports. | |
| **Roles, Services, and Authentication** | Logical separation of required and optional roles and services. | Role-based or identity-based operator authentication. | Identity-based operator authentication. |
| **Physical Security** | Production grade equipment. | Locks or tamper evidence. | Tamper detection and response for covers and doors. |
| **Operational Environment** | Single operator. Executable code. Approved integrity technique. | Referenced PPVs evaluated at EAL2 with specified discretionary access control mechanisms and auditing. | Referenced PPVs plus trusted path evaluated at EAL3 plus security policy modeling. |
| **Cryptographic Key Management** Key management mechanisms: random number and key generation, key establishment, key distribution, key entry/output, key storage, and key revocation. | Secret and private keys established using manual methods may be entered or output in plaintext form. | Secret and private keys established using manual methods shall be entered or output encrypted or with split knowledge procedures. | |
| **Self-Tests** | Power-up tests. cryptographic algorithm tests, software/firmware integrity tests, critical functions tests. Conditional tests. | | |
| **Mitigation of Other Attacks** | Specification of mitigation of attacks for which no testable requirements are currently available. | | |
2.2.2 Requirements for Certification for Various Levels

The security requirements encompass areas related to the design and implementation of a Cryptographic Module are discussed in detail in the following sections.

2.2.2.1 Cryptographic Module Specification

The Cryptographic Module Specification (CMS) describes the Cryptographic Module, including all hardware, software, and firmware components, and delineates the module security policy. The CMS shall define the cryptographic boundary, i.e., the domain that contains the processor(s) and any other elements that might store software or firmware configuration items. The CMS Specification shall:

- Identify the hardware, software, and firmware components of a Cryptographic Module and specify the physical boundaries of the module.
- Identify any hardware/firmware that must be excluded from the cryptographic boundary and provide rational for such exclusion.
- Account for all the physical and logical interfaces and all input/output paths in the Cryptographic Module.
- List all the approved/non-approved security functions that are employed by the module and specify all the modes of operation.
- Define functional and physical block diagrams depicting all the hardware components and various interconnects in the Cryptographic Module. It shall also include a high level flow chart/algorithm for the module resident software/firmware.
2.2.2.2 Cryptographic Module (CM) Ports and Interfaces

The Cryptographic Module (CM) ports and interfaces shall be logically unique although they may share a common physical port. The Cryptographic Module shall contain the following four logical interfaces:

- **Data input interface:** All data inputs shall interface with the Cryptographic Module through the “data input” port. These input data may include, but are not limited to, plain text, cipher text, cryptographic keys, control information, or other CSPs.

- **Data output interface:** All post-processed Cryptographic Module output data shall exit from the “data output” port. Output data may include plain text data, cipher text data, cryptographic keys, control information, or other CSPs.

- **Control input interface:** All Cryptographic Module command and control signals/data shall enter through the “control input” interface. These data may include function calls and manual controls such as switches, buttons, keyboards, etc.

- **Status output interface:** Cryptographic Module status data and signals shall be output through the “status output” interface. These data may include return codes or physical indicators like LEDs and displays.

2.2.2.3 Roles, Services, and Authentication

The Cryptographic Module shall have one Identity-Based Operator (IBO), which shall require authentication. Authentication shall be implemented in software by a normal password mechanism. The IBO shall have exclusive rights to enable/disable the
maintenance state, the error state, etc. Based on the services performed by the Cryptographic Module for each user, there may be different means of authenticating different users.

2.2.2.3.1 Roles

The Cryptographic Module may support the following roles for various operators:

- User Role: In this role the user performs general security services such as cryptographic operations and other approved security functions.
- Crypto Officer (CO) Role: The CO performs more specific functions such as Cryptographic Module initialization or input/output of cryptographic keys, etc.
- Maintenance Role: In the maintenance role the user may perform physical/logical maintenance of the Cryptographic Module. Prior to entering the maintenance mode, all private keys and other CSPs will be nulled before entering/exiting the role.

2.2.2.3.2 Services

A Cryptographic Module shall provide the following operator services:

- Report Status: Output the current status of the Cryptographic Module.
- Perform Self Tests: Initiate and execute the self tests.
- Perform Approved Security Functions: Perform at least one approved security function used in an approved mode of operation.
2.2.2.3.3 Authentication

Different users may assume different roles as described above. Each user role shall require authentication as prerequisite for assuming the desired role. Authentication may be of two types:

- **Role-Based Authentication:** In role-based authentication, the module may require that one or more roles be implicitly or explicitly selected by the operator and shall authenticate the assumption of the selected role. In this mode the module does not need to authenticate an individual for a role.

- **Identity-Based Authentication:** This authentication type requires the requestor to authenticate himself or herself based on various identity recognizing technologies available. Once authenticated, the user can assume a role associated to his/her profile. If the module permits it, an operator can change roles while logged into the module or add another role not previously assigned. Various types of authentication data may be required by a module to implement identity-based authentication such as password, personal identification number (PIN), or biometric authentication combined with any of the above methods. Requirements for such PIN- or password-based authentication techniques are delineated in the NIST PUB 140-2 (National Institute of Standard and Technology, 2002).
2.2.2.4 Finite State Model

The Cryptographic Module shall have several operating states as depicted in Figure 2-5. The state transition diagram includes:

- all operational and error states of a Cryptographic Module;
- the corresponding transitions from one state to another;
- the input events that cause transitions from one state to another; and
- the output events resulting from transitions from one state to another.

![Figure 2-5: States in a Cryptographic Module](image)

2.2.2.5 Physical Security

The Cryptographic Module may implement physical barriers in order to avoid and detect restricted unauthorized physical access to the module and its contents. Therefore, all hardware, firmware, and data components within the cryptographic boundary shall be protected. Different security levels may require stricter physical security requirements. Table 2-3 summarizes physical security requirements for various security levels.
2.2.2.6 Cryptographic Key Management

The FIPS Publication 140-2 places special emphasis on the generation–handling–disposing—i.e., the entire life-cycle—of the cryptographic key used during the ciphering/deciphering process in the module. This key management involves key generation, establishment, distribution, key entry/output, storage, and nullification.

2.2.2.6.1 Random Number Generators (RNGs)

A Cryptographic Module may use random number generators (RNGs). Approved or non-approved RNGs may be used; however, the RNG shall pass the continuous random
number generator test (NIST: Lawrence E Bassham, 2008) in order for its output to be used in the key generation process.

2.2.2.6.2 Key Generation

If the Cryptographic Module is generating keys internally, it shall to comply with certain FIPS requirements. Only approved key generation algorithm/methods shall be used. If intermediate key generation values are output from the module, these values shall be encrypted or under split knowledge procedure.

2.2.2.6.3 Key Establishment

There are different ways of establishing keys within the Cryptographic Module: the Cryptographic Module may establish a key or it may accept an inserted key. Cryptographic Module key establishment shall use only an approved method.

2.2.2.6.4 Key Entry and Output

Cryptographic keys may be entered into the Cryptographic Module or removed from the module. If such entry or removal takes place, it shall occur using either manual or electronic methods such as keyboards or smart cards, etc. If the entire key is being entered, its value shall be verified during entry using key entry tests specified in the Standard.
2.2.2.6.5 Key Storage

The cryptographic keys can be stored within the Cryptographic Module in either plain-text or ciphered format. These keys shall not be accessible to unauthorized operators.

2.2.2.6.6 Key Zeroization

A Cryptographic Module and the CSPs contained therein shall be protected in case of a security breach. For this reason, the CSPs and the cryptographic key shall be nullified (zeroed) in case of physical tampering of the module or during power up. The key shall also be zeroized during certain states like maintenance so that the security of the data and the key is not compromised.

2.2.2.7 Cryptographic Boundary

The cryptographic boundary is defined in physical terms as the Cryptographic Module hardware and the embedded firmware.
Chapter 3
System Architecture and Interface

This chapter provides a detailed description of the system architecture and its design. The chapter will cover topics such as system modules, its interface (both hardware and software), and finally presents in detail the firmware developed. The chapter also briefly covers the module functions.

3.1 System Overview

The system overview and the detailed description of the end-to-end process are described in this section. The transmission of data from the sensor node to the HUB and back from the HUB to individual sensor nodes is divided into five steps. These steps involve data inputs, executing various algorithms on the data, and output of the processed data. The basic and detailed process steps are depicted in Figures 3-1 and 3-2, respectively.
Step #1: Master MSP sends raw data to the Cryptographic Module.

Step #2: Cryptographic Module ciphers the data using cryptographic algorithms.

Step #3: Cryptographic Module then sends the ciphered data back to the Master MSP.

Step #4: The Master MSP sends the ciphered data to the Dust Networks Radio.

Step #5: The Radio then transmits the data over a wireless channel.

If the above system is to put into a simplified block diagram it can be represented in the following Figure 3-2:

Figure 3-2: Detailed system architecture depicting the four modules

As depicted in Figure 3-2, the system consists of three main subsystems, which in turn consist of a number of modules including: the S5NAP node, Dust Networks wireless network, and the receiver terminal.
The $S^5$NAP node consists of the Sensor Subsystem, Cryptographic Module, Master MSP430 and Dust Radio. The Dust Network consists of a number of Dust Radios passing information in an *ad hoc* manner. Further details of the Dust Network are beyond the scope of this thesis. The receiver element comprises of the Master terminal, which is essentially a connector program running on a PC and the Cryptographic Module (same as that at the transmitter $S^5$NAP node).

### 3.1.1 Sensor Subsystem

The Sensor Subsystem measures the vibrations on the equipment it is mounted on. These vibrations in the $S^5$NAP device are measured via an integrated charge-mode accelerometer. The analog output of the accelerometer is input to two different components: (i) the waveform acquisition circuitry and (ii) the continuous vibration monitoring circuitry. Table 3-1 lists Sensor System details.

<table>
<thead>
<tr>
<th>System</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
<td>The Sensor Subsystem &amp; Energy Harvester</td>
</tr>
<tr>
<td><strong>Abbreviation</strong></td>
<td>Sensor Subsystem</td>
</tr>
<tr>
<td><strong>Main Components</strong></td>
<td>NA</td>
</tr>
<tr>
<td><strong>Point of Contact</strong></td>
<td>NA</td>
</tr>
<tr>
<td><strong>Developer</strong></td>
<td>Impact-RLW Systems, Inc</td>
</tr>
</tbody>
</table>
3.1.2 Master MSP430

The Master MSP430 is responsible for executing the S\(^5\)NAP firmware. The firmware’s primary responsibilities are to gather vibration data from the ADC; report status and data information wirelessly using the DN2140; respond and reconfigure the reporting and acquisition settings on-demand; and perform over-the-air wireless firmware updates. The firmware shall adhere to a strict power budget since the S\(^5\)NAP is a battery operated device. The Master is also responsible for maintaining the SPI connection with the Cryptographic Module and interrupting (waking) the Cryptographic Module when data needs to be ciphered/deciphered. Table 3-2 contains details of the S\(^5\)NAP.

Table 3-2: Details of the Master MSP430 chip.

<table>
<thead>
<tr>
<th>System</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Master controller for S(^5)NAP firmware and Cryptographic Module</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Master</td>
</tr>
<tr>
<td>Main Components</td>
<td>TI MSP430F1611 processor, Connector terminal</td>
</tr>
<tr>
<td>Point of Contact</td>
<td>Eric Humenay, Software Engineer, Impact-RLW Inc.</td>
</tr>
<tr>
<td>Developer</td>
<td>Impact-RLW Systems</td>
</tr>
</tbody>
</table>

3.1.3 FIPS MSP430

The Cryptographic Module, which is comprised of the FIPS MSP430 processor, is primarily responsible for encrypting data acquired by the sensor subsystem and decrypting messages received from the wireless mesh network. All data and messages are provided to the FIPS MSP30 processor by the Master MSP430. After the FIPS MSP430
has encrypted or decrypted the information it is then passed back to the Master MSP430. Thus, the FIPS MSP430 acts as a Cryptographic Module that is responsible only for the ciphering and deciphering of data and for producing the necessary keys required. The Master MSP430 is the only device it is connected to in the system. The Table 3-3 contains details of the Cryptographic Module.

<table>
<thead>
<tr>
<th>System</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>FIPS-Certified Cryptographic Module</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Crypto</td>
</tr>
<tr>
<td>Main Components</td>
<td>TI MSP430F1611 processor</td>
</tr>
<tr>
<td>Point of Contact</td>
<td>Jesal Kanani</td>
</tr>
<tr>
<td>Developer</td>
<td>The Pennsylvania State University</td>
</tr>
</tbody>
</table>

3.1.4 Dust DN2140 Radio

The Dust DN2140 radio is a commercial-off-the-shelf (COTS) proprietary IEEE 802.15.4 wireless mesh networking solution provided by Dust Networks. The DN2140’s communication protocol is proprietary in nature and thus is treated as black box in the S5NAP system. The Table 3-4 provides information on the DUST Radio.

<table>
<thead>
<tr>
<th>System</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>DUST DN2140 Radio</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>DN2140</td>
</tr>
<tr>
<td>Main Components</td>
<td>NA</td>
</tr>
<tr>
<td>Point of Contact</td>
<td>NA</td>
</tr>
<tr>
<td>Developer</td>
<td>Dust Networks</td>
</tr>
</tbody>
</table>
3.2 System Interface

There are both hardware and software interfaces within the system. Of particular interest in this work is the interface between the Master and the Crypto MSP processors. This hardware interface consists of a serial bus and software routines for handshaking protocols. Other important interfaces involving the MSP430 processors or the Dust radio are the interfaces between the connector (PC) and the Dust Manager, the Connector and the Crypto USB dongle. These interfaces are depicted in Figure 3-3.

![Figure 3-3: End-to-end system between sensor nodes](image)

The modularity of the system architecture partitions functions to interdependent modules and interfaces enabling an end-to-end communication system. The system interfaces are depicted in Figure 3-4.
3.2.1 Cryptographic Module to Master MSP420 Interface

The interface between the Cryptographic Module and the Master MSP430 is where a majority of traffic will be generated and distributed throughout the system. The hardware logical/functional interfaces are described below and shown in Figure 3-5.

**Figure 3-5: Interface between Cryptographic Module and Master MSP430 chip**

3.2.1.1 Cryptographic Module to Master MSP430 Hardware Interface
The Crypto–Master interface uses a Serial Peripheral Interface Bus (SPI) bus to transfer data between the two modules. The SPI bus is a synchronous serial data link. The Master and the Crypto MSP transfer data back and forth using the SPI link. The interrupts provided are used to activate the chips or alert of an impending data transfer. The data transfer is synchronized by the SPI_CLK and is transmitted by the Master MSP to the Cryptographic Module. The SPI link and the corresponding pin connections between the two MSP430F1611 processors are shown in Figure 3-6.

![Figure 3-6: Pin interface between the Master and Crypto MSP430F1611s](image)

3.2.1.2 Cryptographic Module to Master MSP430 Logical Interface

The Master is the only element in the system that shall have contact with the Cryptographic Module. Thus, it is important to define this interface in detail. As the vibration data is collected by the Master from the Sensor Subsystem, this data must be
inserted into packets and ciphered before sending it out on the Dust Network and ultimately to the receiving hub. The Master collects data from the Sensor Subsystem and sends the data to the Cryptographic Module for ciphering. The Cryptographic Module accepts the data, ciphers the data packets, and returns the processed data to the Master MSP processor. However, if the Master has some data that is in ciphered form and needs deciphering, it will forward this packet to the Cryptographic Module as well, which decipher the packet and return it to the Master.

In case the Cryptographic Module does not possess the required key to perform the ciphering/deciphering of the incoming packets, it will initiate a key exchange process in association with the Master. The ciphering/deciphering functions will only be resumed once the appropriate keys have been established and exchanged. Although there is an implied simplicity in the process, a number of complex data exchanges occur prior to and during the entire process. The entire key exchange process is explained in detail in Chapter 4.

### 3.2.2 Master MSP430 to Dust Radio DN2140 Interface

The interface between the Master MSP430 processor and the DN2140 Radio is implemented in S5NAP firmware. The packets received by the Master from the Cryptographic Module are forwarded to the Dust Radio for transmission through the Dust Network. Concurrently, the incoming packets from the Network are passed on to the Master by the Dust Radio. Hence, this is a two-way communication interface. It is worth mentioning that the Master MSP430 can receive information from and configure the
DN2140 through the DN2140 serial command interface. Local communication (packets that are never sent out to the wireless network) between the Master MSP430 and DN2140 are not sent to the Cryptographic Module. When a DN2140 radio sends a message to the Dust Manager, the DN2140 Radio is not informed whether or not the message was successfully received at the message’s destination. Thus, Dust’s wireless mesh networking protocol does not support end-to-end acknowledgements of successful message transfers.

The DN2140 is an entity of the Dust Networks and is not within the scope of this document. It should be understood that once the packet is transmitted to the Radio, it is the function of the Dust Network to route it properly to the intended destination.

### 3.2.2.1 Master MSP430 to Dust Radio DN2140 Hardware Interface

Communication between the MSP430 and the DN2140 occurs via a Universal Asynchronous Receiver/Transmitter UART interface at 9600 baud. All messages between the MSP430 and DN2140 are verified by calculating a frame checksum (FCS) based on the 16-bit FCS computation method. The DN2140 checks the FCS on all messages received from Master MSP430 and drops packets containing FCS errors. There is no mechanism for the DN2140 to advise the Master MSP430 that a packet has been discarded. All packets are in big-endian order.
3.2.2.2 Master MSP430 to Dust Radio DN2140 Logical Interface

The interface between the Dust Radio and the Master is a unique design for this system and is described in the following paragraphs. The Master sends a packet to the Dust Radio in a specific format, which contains the ciphered data packets along with an identifier field and the heartbeat message (this is also encrypted and can thus be treated as part of the payload). In addition, the Master also sends a CRC checksum for each packet attached to it for error detection at the Dust Radio. This CRC checksum, however, is not forwarded by the Radio into the network and is clipped off at the Radio. Thus, this CRC is used only to check the packet integrity between the Master and the Radio.

When the Dust Radio receives a packet from the network, it forwards this packet to the Master. The Dust Radio also relays the originating address to the Master. This information is useful for the Master when it is forwarding the packet to the Cryptographic Module. It should be noted that every time a packet passes through a Dust Radio into the network, the DN2140 appends a header called the Dust Header to the packet. This header contains basic routing information, destination address, hop information, etc. Thus, when a packet is received by a Dust Radio from the network, it simply clips off this header and passes on the rest of the packet to the module next in line.

It should be noted here that when a packet is sent from the Cryptographic Module to the Master, the packet contains the destination address. However, when this packet is being further forwarded to the Dust Radio, this destination address is clipped off from the packet (to save some useful bits as a maximum of 80 bytes can be transferred to the Dust Radio at a time). When a Dust Radio receives a packet from the Master, it inspects the
packet but finds no information of the intended destination for the packet. This problem can be solved either by keeping the destination address on the packet or sending command bits to the Dust Radio before transmitting the actual packet. This needs to be addressed in the future implementations of the system.

In addition, as mentioned above, the Dust Radio needs to transmit the information to the Master to identify the packet origin every time it forwards a packet from the network to the Master. Will the Dust radio perform this function voluntarily after a packet is forwarded to the Master or will the Master ask for this information explicitly? This also needs to be addressed in the future implementations of the system.
Chapter 4

Implementation

This chapter is presented in two segments. The first deals with various functions implemented on the Cryptographic Module MSP430F1611 processor. These functions and their algorithms are described in detail. The second segment focuses on the entire system and how packets and data are communicated from one module to the other. For this, a new and project-specific system was designed and is presented in this chapter.

4.1 Cryptographic Module

The Cryptographic Module performs a series of operations on the incoming data packets as prescribed by the requirements for FIPS certification. Listed below are some of the functions that will be performed by the Cryptographic Module on the data packets provided by the Master MSP430 processor. To maintain the symmetry between the Cryptographic Modules at both the sensor node and the connector end, the same module will be installed. The functions performed by are described in the following paragraphs.

4.1.1 Ciphering/Deciphering

There have been different implementations of the AES modified to suit certain constraints. These constraints could be speed, memory, or power as it is in our case. In certain cases (Law, Doumen, & Hartel, 2006), there have been implementations of the AES on the MSP430 processor as well. They have implemented the code on OpenSSL
and modified it heavily. Typically the AES gives the best performance in Output Feedback Mode, in which it takes 2137 clock cycles to encrypt blocks of 128-bits and takes up to 12860 bytes of FLASH memory for the code. Based on their size optimization of AES code, it is now implemented in 12616 bytes of FLASH and 70 bytes of RAM to cipher the same amount of block-data.

In other implementations (Vitaletti & Palombizio, 2007) the focus is on compact implementation, which also yields good results in terms of space utilization. However, this hampers the performance in terms of time of execution of the code. Figure 4-1 shows the implementation in which both the key and data are used from the RAM for the AES ciphering, whereas the code is executed from the FLASH

\[\text{Figure 4-1: Ciphering process using AES algorithm}\]

The input and output for the AES algorithm each consist of sequences of 128 bits. The Cipher Key for the AES algorithm is a sequence of 128, 192, or 256 bits. The Cryptographic Module’s processor will accept data from the Master MSP430 via the SPI
bus. The plaintext data will be stored in the RAM of the Cryptographic Module’s MSP430 processor and ciphered in segments of 128 bits each. The AES algorithm consists of four important rounds. This involves shifting, substituting, key expansion, and column exchange. The ciphered data will be then stored in the memory and transferred to the Master MSP once the entire block of data has been ciphered. A detailed description of the AES ciphering algorithm is presented in the Appendix.

4.1.2 Key Establishment

In a technique of key establishment (Diffie & Hellman, 1976) it was proposed that a new method of public key exchange can be used where any new entity can join the system without having to possess pre-shared keys with other entities in the system. In contrast different schemes like (Du, Deng, Han, Chen, & Varshney, 2004) have described a random key pre-distribution scheme that uses deployment knowledge. With this kind of knowledge, only a fraction of keys need to be carried by a node as compared to other similar mechanisms (Eschenauer & Gligor, 2002) and (Chan, Perrig, & Song, 2003).

Since the system uses the AES algorithm for encryption of the data generated by the sensor node, it requires a ciphering key. This ciphering key can be of variable length, which in turn decides the strength of the ciphering. (For programming and debugging simplicity, initially an AES algorithm accepting a key length of 128 bits is implemented). This key needs to be stored at the transmitting as well as the receiving end for adequate ciphering and deciphering. Thus, there is a need for a key exchange and key establishment process, for which we have used the Diffie–Hellman process.
In the Diffie–Hellman process, a prime number $p$ and a generator $g$ is known to all the nodes of the network. Suppose Alice and Bob want to agree on a shared symmetric key. Alice and Bob and everyone else already know the values $p$ and $g$. Alice generates a random private value $a$ and Bob generates a random private value $b$. Now, each computes a public value: $g^x \mod p$, where $x$ is the private value generated at each node. So,

Alice’s public value: $g^a \mod p$ and

Bob’s public value: $g^b \mod p$.

These public values are sent to each other respectively.

Finally, Alice computes $g^{ab} \mod p = (g^b \mod p)^a \mod p$ and

Bob computes $g^{ba} \mod p = (g^a \mod p)^b \mod p$.

Alice and Bob now have $g^{ab} \mod p$ as their shared symmetric key. Any eavesdropper would know $p$, $g$, and the two public values $g^a \mod p$ and $g^b \mod p$. If only the eavesdropper could determine either $a$ or $b$, she could easily compute the resulting key. Determining either $a$ or $b$ from the information she has, however, is computationally infeasible for suitably large $p$, $a$, and $b$; it is known as the “discrete logarithm problem”.

---

[Diagram: Diffie–Hellman key exchange process]
4.1.3 Zeroization

Zeroization a process that involves resetting all the CSPs in case of any tampering with the Cryptographic Module. At Level 2, automatic zeroization is not required. Thus, a manual switch which resets the module will power up the Crypto MSP430 processor, which in turn can zeroize all the CSPs. After zeroization, upon startup, the Cryptographic Module will perform self integrity checks and run known answer tests (KATs). The above mentioned process is implemented by overwriting a section of or the entire RAM of the processor by zeroes. A rough algorithm for the same is shown in Figure 4-3 below.

```
Key zeroization (void)
{
  If (security_breach || power reset)
  {
    RAM_pointer = Start of RAM address;
    for( i= Start of RAM; i<= End of RAM; i++)
    {
      *RAM_pointer = 0;
      RAM_pointer++;
    }
  }
  return;
}
```

Figure 4-3: Algorithm for zeroization of RAM

As seen in Figure 4-3, the zeroization process occurs either during power up cycles or during a physical security breach. In both cases all the CSPs are lost.
4.1.4 Known Answer Tests

Known Answer Tests (KATs) are used for checking the integrity of the software or a particular algorithm that has been implemented. For testing the integrity of the software at power up, we need to have a test bench set up that will check if the software (algorithms) is working correctly. This test bench will be run on the algorithm and the outputs obtained will be compared with the known correct outputs. A match indicates a noncorrupt FLASH. The KATs are stored in the flash memory on the MSP430. Since upon zeroization only the RAM has to be erased, the software remains intact in the flash memory. Thus, on startup the processor will pick up the test inputs from the memory and run them through the AES and the authentication programs. The output of these will then be compared to a pre-calculated correct output. If they match, the Cryptographic Module will go ahead with its next task. If not, then it will go into an error state.

```c
main (void)
{ :
  If (power_up)
    KAT( )
  :
}

KAT(void)
{
  Input_to_AES = ‘test bench’;
  Test_bench_result = Output_from_AES;
  if ( Output_from_AES == pre_calculated_result_for_test_bench)
    printf(“ KAT passed”);
  else
    printf(“ KAT failed. Enter Maintenance state”);
  return;
}
```

Figure 4-4: Algorithm for Known Answer Tests (KATs)
4.1.5 Startup Tests

The startup tests are aimed at ensuring there has been no tampering with the software. To confirm this, startup tests are run to make sure the software that is stored in the memory of the Cryptographic Module MSP430 processor is uncompromised. This can be achieved by running a checksum algorithm and comparing the final answer to a stored one. During startup, the entire memory of the processor is considered as one big block of data, and we have to check if that block of data is still the same as it was before shutdown. If the result of the checksum matches the one already stored, then the software is deemed to not have been tampered with and the processor can go ahead with its next task.

4.2 Data Transfer in System

Table 4-1: Identifier bits and the corresponding payload in the packet

<table>
<thead>
<tr>
<th>Identifier Field</th>
<th>Direction</th>
<th>Value of Data Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXXX111</td>
<td>Master</td>
<td>Crypto</td>
</tr>
<tr>
<td>XXXXX100</td>
<td>Master</td>
<td>Crypto</td>
</tr>
<tr>
<td>XXXXX000</td>
<td>Master</td>
<td>FIPS</td>
</tr>
<tr>
<td>XXXXX001</td>
<td>Master</td>
<td>Crypto</td>
</tr>
<tr>
<td>XXXXX010</td>
<td>Master</td>
<td>Crypto</td>
</tr>
<tr>
<td>XXXXX011</td>
<td>Master</td>
<td>Crypto</td>
</tr>
<tr>
<td>XXXXX101/110</td>
<td>NA</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

As mentioned before, a unique packet flow algorithm is designed for the system in this project. Figure 4-5 depicts the flowchart of the algorithm. In the flow chart depicted in Figure 4-5, the path of a typical packet containing data can be seen. As the packet passes from one module to the other, data encapsulation occurs.
Figure 4-5: Flowchart depicting the flow of data packets in the
Step 1: \[\text{Bxx} \quad \text{xxxx111} \quad \text{Hmsp} \quad \text{Plaintext Data}\]

Where we define Bxx to be the address of the Receiver mote at the Master terminal; xxxxx111 indicates that the data in this packet is plaintext data that needs ciphering; and Hmsp is the mandatory Header of the Master MSP in each packet. This is the first step towards building a new key. It initiates the key exchange process. In this step, the Master MSP needs to cipher some data, so it will pack this data in the packet as shown and send the packet to the Cryptographic Module.

Step 2: \[\text{Bxx} \quad \text{xxxx111} \quad \text{Hmsp} \quad \text{Plaintext Data}\]

The Cryptographic Module receives the packet sent to it in Step 1 and looks into the header (xxxxx111) and the address (Bxx). Thus, it realizes that the packet contains plaintext data and that it needs to cipher this plaintext data with a key that corresponds to the mote with address “Bxx”. Since the Cryptographic Module does not have a key to cipher the plaintext data in this packet sent by the Master MSP, it will send it back to the Master in Step 2. No change to the packet is done here by the Cryptographic Module.

Step 3: \[\text{Bxx} \quad \text{xxxx000} \quad \text{Hmsp} \quad \text{Empty}\]

In this step, xxxxx000 indicates that the packet is an empty envelope for inserting key material. The Master MSP is expecting from the Cryptographic Module a ciphered version of the packet it sent in Step 1. However, it receives the same non-ciphered packet back from the Cryptographic Module and thus realizes that the Cryptographic Module does not have the key to do the ciphering for the node in consideration. Thus, it generates
three (can be changed) packets, which can be considered as empty envelopes into which the Cryptographic Module will insert its half of the key. Thus, in Step 3, the Master MSP sends these empty packets meant for key insertion to the Cryptographic Module.

---

**Step 4:**

| Bxx | xxxxx001 | Hmsp | Part of Key | Bxx | xxxxx010 | Hmsp | Part of Key | Bxx | xxxxx011 | Hmsp | Part of Key |

Bxx is the address of the Receiver mote at the master terminal; xxxxx001 indicates that this is Packet #1 containing the key material in data field; xxxxx010 indicates that this is Packet #2 containing the key material in data field; xxxxx011 indicates that this is Packet #3 containing the key material in data field.

As the Cryptographic Module receives the empty packets for key insertion, it generates half of the key as per the Diffie–Hellman algorithm and inserts it into these envelopes. Thus, the total length of half of the key it generates is 128 bytes, and each empty envelope sent by the master can accommodate only 50 bytes of data; then the Cryptographic Module will insert the first 50 bytes in Packet #1 (with header xxxxx001), the next 50 bytes in Packet #2 (with header xxxxx010) and the remaining 28 bytes in Packet #3 (with header xxxxx011). In Step 4, these key containing packets are sent to the Master MSP, which is expecting them.

---

**Step 5:**

| xxxxx001 | Hmsp | Part of Key | xxxxx010 | Hmsp | Part of Key | xxxxx011 | Hmsp | Part of Key |

In Step 5, the Master MSP receives the key material filled envelope packets and sends these to the Dust Radio for transmission out to the Dust network. However, there is a subtle difference between the packet the Master MSP receives in Step 4 and the packets it
sends out in Step 5. The Master MSP strips off the address field (Bxx) from the packets before sending them to the Dust radio. The reason for this stripping of the address field is that, at the receiver end, the Dust radio does not require knowledge of its own address but it needs to know where the packet is coming from. In other words, at the receiver, the Dust radio already knows its mote address (Bxx) and so keeping this address (Bxx) in the packet is not of any use. On the contrary, the receiver mote needs to know where the packet is coming from. Thus, it may sound logical to strip of the receiver address (Bxx) and instead put your own transmitter mote address (Axx). Thus, when this packet is now received at the receiver end and the Dust radio passes it to the Master terminal and finally to the Cryptographic Module, it will know where the packet it coming from. However, this same information can be achieved from the Dust radio at the receiving end separately as well (as shown in Step 7) and so we save some bandwidth by not sending the transmitter address (Axx) explicitly in the packet. This is explained further in Step 7.

**Step 6:**

\[
\begin{array}{c}
H_{\text{dust}} \\
xxxxx001 \\
H_{\text{msp}} \\
\text{Part of Key} \\
T_{\text{dust}}
\end{array}
\]

\(H_{\text{dust}}\) is the header attached by the Dust radio, \(T_{\text{dust}}\) is the trailer attached by the Dust radio, and \(xxxxx001\) is the identifier that indicates that this is Packet #1 containing the key in data field. As the packets are sent to the Dust radio by the Master MSP in Step 5, these packets need to reach the receiving end through the Dust network. For this, the Dust radio attaches a header and trailer to the received packet and then transmits it out into the Dust network. This header/trailer contains useful information that will be required by the network to determine the route of the packet through the system, the destination address,
and other such useful information. At the receiving end, these same packets with the headers and trailers are relayed to the Dust radio, which then forwards it further after some processing.

---

**Step 7:**

```
Axx xxxxx001 Hmsp Part of Key
```

Here, xxxxx001 is the identifier that indicates that this is Packet #1 containing the key in data field and Axx is the address of the transmitter mote. Each mote in the network has a unique address. The Dust radio receives packets from the network that contain the header/trailers. Thus, in Step 7 the radio strips off these headers and trailers from the packets and forwards the rest of the packet to the Master terminal/connector. Along with the packet, it also sends to the Master terminal/connector the address of the mote from which the packet is received. Thus, the receiver mote now knows where the packet has come from. Since this information could be gathered from the receiver Dust radio as we just mentioned, the transmitter address (Axx) was not included in the packet. This supports the argument in Step 5 for not attaching this transmitter address (Axx) in the packet thus saving some bandwidth.

---

**Step 8:**

```
Axx xxxxx001 Hmsp Part of Key
```

The Master terminal/connector receives two things from the Dust radio: the packets and the transmitter address field (Axx in this case). The Master connector appends this address field to the packet and creates a packet as shown in Step 8. Thus, the Master
connector sends 3 packets to the Cryptographic Module that contains key exchange material. The Master makes a note of this and proceeds to Step 9.

---

**Step 9:**

| Axx | xxxxx000 | Hmsp | Empty |

Here, xxxxx000 indicates that the packet is an empty envelope for inserting key material and Axx is the address of the transmitter mote. As the Master Connector forwards the key containing packets in Step 8 to the Cryptographic Module, it understands that the other end has initiated a key exchange process and that it requires sending out the other half of the key to complete the entire key exchange process. This other half can be generated only by the Cryptographic Module and so just as in Step 3; the Master Connector sends 3 packets, which can be considered as empty envelopes in which the Cryptographic Module will insert its half of the key.

---

**Step 10:**

| Axx | xxxxx001 | Hmsp | Part of Key |
| Axx | xxxxx010 | Hmsp | Part of Key |
| Axx | xxxxx011 | Hmsp | Part of Key |

Here, Axx is the address of the Transmitter mote, which now happens to be the receiver, xxxxx001 indicates that this is Packet #1 containing the key material in data field, xxxxx010 indicates that this is Packet #2 containing the key material in data field, and xxxxx011 indicates that this is Packet #3 containing the key material in data field.

As the Cryptographic Module receives the empty packets for key insertion, it generates half of the key as per the Diffie–Hellman algorithm and inserts it into these envelopes. Thus, the total length of half of the key it generates is 128 bytes, and each empty envelope sent by the master can accommodate only 50 bytes of data; then, the
Crypto MSP processor will insert the first 50 bytes in Packet #1 (with header xxxxx001), the next 50 bytes in Packet #2 (with header xxxxx010) and the remaining 28 bytes in Packet #3 (with header xxxxx011).

Step 11:  

The Master MSP receives the key material filled envelope packets and, in Step 11, it sends these to the Dust radio for transmission out to the Dust network. However, just like in Step 5, the Master MSP strips off the address field (Axx) from the packets before sending them to the Dust radio. The reasoning is also the same as given in Step 5 and Step 7, i.e., bandwidth conservation.

Step 12:  

H\text{dust} is the header attached by the Dust radio, T\text{dust} is the trailer attached by the Dust radio, and xxxxx001 is the identifier that indicates that this is Packet #1 containing the key in data field. As the packets are sent to the Dust radio by the Master MSP in Step 11, these packets need to reach the receiving end through the Dust network. For this, the Dust radio attaches a header and trailer to the received packet and then transmits it out into the Dust network. This header/trailer contains useful information that will be required by the network to decide the route of the packet through the system, the destination address and other such useful information. At the receiving end, these same packets with the headers and trailers are relayed to the Dust radio, which then forwards it further after some processing.
Step 13:  

Here, xxxx001 is the identifier indicates that this is Packet #1 containing the key in data field and Bxx is the address of the receiver mote, which now can be addressed as the transmitter. The Dust radio receives packets from the network, which contains the header/trailers. Thus, in Step 7, the radio strips off these headers and trailers from the packets and forwards the rest of the packet to the Master terminal/connector. Along with the packet, it also sends the Master terminal/connector the address of the mote from where the packet is received. Thus, the receiver mote now knows from where the packet has come.

Step 14:  

The Master terminal receives two things from the Dust radio: the packets and the transmitter address field (Bxx in this case). The Master terminal appends this address field to the packet and creates a packet as shown in Step 14. It forwards this packet to the Cryptographic Module, thus completing the entire key exchange process. It is important to notice here that the Master terminal does not send out empty envelopes again (which is its reaction whenever it receives packets containing key material) because in this case the Master terminal knows it was the one who initiated the key exchange process and so the packets it just received are in response to the packets it had sent out previously.
Chapter 5

Testing

This chapter gives an insight on how the software developed and the communication protocol was tested. It briefly describes the experimental setup and the hardware and software used for developing and testing the code. Some of the screenshots are attached to display the testing outputs and results.

5.1 Introduction

Development and testing of the various algorithms for the Cryptographic Module were very important tasks. The entire development of the firmware was done using development kits available for general purpose development by Texas Instruments. Software was developed in the IAR Embedded Workbench, a product of IAR Systems. In-circuit emulators were used to program the chip and to perform in-circuit debugging of the firmware.

5.2 Specifications

The technical specifications of the software and hardware used for development and testing are provided below.

Development Kits: We used the 64-pin MSP-TS430PM64 target board specifically for our 64-pin MSP430F1611 microprocessor chips. This is a ZIF socket target board used to
program and debug in-system MSP430 processor using a JTAG interface or SPY-BI wire which is a serialized JTAG protocol developed by Texas Instruments.

---

**Programming Interface:** We used the MSP-FET430UIF programmer to program the MSP430 processor chips in the target boards. The MSP-FET430UIF is a powerful flash emulation tool, which includes a USB debugging interface used to program and debug the MSP430 in system through the JTAG interface. The USB debugging interface (MSP-FET430UIF) connects a flash-based MSP430 MCU to a PC for real-time, in-system programming and debugging.

**IAR Embedded Workbench for MSP430:** We used the IAR Embedded Workbench to develop our firmware using its C compilers. The IAR software with its optimizing C/C++ compiler provides extensive support for devices in MSP430X family and generates very compact and efficient code. It also outputs a binary file in a `.a43` or `ihex` format that can be used to download the firmware into the MSP430 processors using different target
boards. The software provides easy in-circuit debugging of the code on the processor and has many useful libraries for standard functions that can be used.

5.2 Code Development

The software for the entire packet-handling protocol and various algorithms for the Cryptographic Module were implemented in embedded C using the IAR Embedded Workbench and the emulator and target boards. The two target boards were connected using an SPI interface, with one programmed to act as the Cryptographic Module and the other as the Master MSP430 processor. The arrangement is as shown in Figure 5-3 below.

![Experimental set-up for emulating Master-Crypto function](image)

As seen in Figure 5-3, the interface is a replica of what is going to be on the actual hardware as shown in Figure 3-6. The two chips were programmed alternatively using the
FET emulator and then each of the dev kits was powered using the USB. However, every time in-circuit debugging was needed, one of the development kits would be run in steps to check the actual sequence of instructions. The entire experimental setup can be seen in Figure 5-4.

![Experimental setup](image)

**Figure 5-3: Experimental setup**

The two development kits are interfaced and one of them is powered using an external power supply, while the other one is being debugged via the USB flash emulator connected to the PC. On the PC is running the IAR Embedded Workbench software.

A snapshot of the IAR software is given in the Figure 5-5. The IAR has an I/O terminal where one can print statements while executing the code. Thus, while the code is
running on the MSP430 processor, the print statements will be executed at the right instructions to show the status and position of the code. This helps while debugging or implementing logical checking of variable values at various points in the code. Figure 5-5 is a screenshot when the code implemented on the Master–Crypto is the two-way transfer of a packet on the SPI link. The print statements are self-explanatory.

Figure 5-4: Screenshot of the IAR with print statements
The entire firmware that is being developed will finally be tested and implemented on the boards for the S\textsuperscript{5}NAP, which are developed by Impact-RLW Inc. These boards & the support boards is an Impact-RLW proprietary and hence will not be discussed here.

5.3 Test bench inputs

As mentioned earlier, the Cryptographic Module performs the required functions based on the value of the identifier bits in the packet. Thus if the identifier bits have a value ‘07’ in hex, the Cryptographic Module will cipher the contents of that packet and so on (please refer Table 4-1).

Below are the test bench inputs given to the cryptographic module along with the outputs. The important thing to note is that the reason for this input–output text here is purely to demonstrate the capability of the Cryptographic Module to distinguish between packets based on the identifier bits and perform the related functions. The actual functions of ciphering/deciphering and key generation are not demonstrated here. The Cryptographic Module performs a ‘divide by 2’ function if the packet requires ciphering or ‘multiply by 2’ when it requires deciphering. Similarly, the key generated and inserted in an empty envelope is not the actual key but just a string of zeroes.

**Test Bench 1:** The Test bench 1 shows the packet sent to the Cryptographic Module has the identifier bits ‘07’. Thus the cryptographic Module will cipher the data in this packet. It should also be noted that the identifier bits in the outgoing packet are changed to ‘04’.
Input: ff 07 ff f8 fb dd 59 bb fd 6d 47 97 fc c5 ff ef b3 cf 7f fc df fa f5 ff e7 3f df e7 1d ef ad 7f d3 ff b1 5f d2 ff f9 ef f6 dd fe f7 ef f7 8b ec fd bd ba ff fa 7f 97 fe 6a 7f fe b6 d7 ef 3c ef 01 ed da 6b be b7 fe 7f e7 9a fb dd ef bf be f5

Output: fe 04 fe f0 f6 ba b2 7e fa da 8e 2e f8 8a fe de 66 9e fe f8 be f4 ea fe ce 7e be ce 3a de 5a fe a6 fe 62 be a4 fe f2 de ec ba fc ee de ee 16 d8 fa 7a 74 fe f4 fe f4 2e fc d4 fe fc 6c ae de 78 de 02 da b4 d6 7c 6e fe ce 34 f6 ba de 7e 7c ea

Test Bench 2: The Test bench 2 shows the packet sent to the Cryptographic Module has the identifier bits ‘04’. Thus the Cryptographic Module will decipher the data in this packet. The identifier bits in the outgoing packet are changed to ‘07’.

Input: ff 04 ff f8 fb dd 59 bb fd 6d 47 97 fc c5 ff ef b3 cf 7f fc df fa f5 ff e7 3f df e7 1d ef ad 7f d3 ff b1 5f d2 ff f9 ef f6 dd fe f7 ef f7 8b ec fd bd ba ff fa 7f 97 fe 6a 7f fe b6 d7 ef 3c ef 01 ed da 6b be b7 fe 7f e7 9a fb dd ef bf be f5

Output: 7f 07 7f 7c 7d 6e 2c 5d 7e 36 23 4b 7e 62 7f 77 59 67 3f 7e 6f 7d 7a 7f 73 1f 6f 73 0e 77 56 3f 69 7f 58 2f 69 7f 7c 77 7b 6e 7f 7b 77 7b 45 76 7e 5e 5d 7f 7d 3f 4b 7f 35 3f 7f 5b 6b 77 1e 77 00 76 6d 35 5f 5b 7f 3f 73 4d 7d 6e 77 5f 5f 7a
**Test Bench 3:** The Test bench 3 shows the packet sent to the Cryptographic Module has the identifier bits ‘00’. Thus the Cryptographic Module will stuff half of the key in this packet. The identifier bits in the outgoing packet are changed to ‘01’.

**Input:**
```
ff 00 ff f8 fb dd 59 bb fd 6d 47 97 fc c5 ff ef b3 cf 7f fc
df fa f5 ff e7 3f df e7 1d ef ad 7f d3 ff b1 5f d2 ff f9 ef
f6 dd fe f7 ef f7 8b ec fd bd ba ff fa 7f 97 fe 6a 7f fe b6
d7 ef 3c ef 01 ed da 6b be b7 fe 7f e7 9a fb dd ef bf be f5
```

**Output:**
```
ff 01 ff f8 fb dd 59 bb fd 6d 47 97 fc c5 ff 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```
Chapter 6
Conclusion and Future Work

The ultimate aim of the project is to deploy FIPS-certified sensor nodes on Navy ships. This chapter provides concluding remarks and proposes some of the future work that can be worked on towards achieving that goal.

6.1 Conclusion

In our work, we proposed an efficient security framework for data generated by sensor nodes and their subsequent transmission over unsecure wireless channels. We have suggested a method for encapsulating packets as they pass from one module to the other and thus make the system as transparent as possible to each entity involved in the communication protocol. Implementing this method will facilitate any other type of implementation between two parallel MSP430 processors. Also, as there is room for future expansion, the system is quite flexible.

The other aspect of the project that required the development of NIST-approved codes for the cryptographic module also was completed successfully. Various codes for algorithms such as AES, Diffie–Hellman, KATs, etc., have been coded in higher level or assembly language and tested. Power optimization also has been achieved by optimizing the code for reduced instructions and loop-elimination algorithms. The cryptographic module thus developed can be deployed to provide strict security services, and yet be very power efficient for battery-operated or energy-harvesting devices.
6.2 Future Work

The following lists some of the future work that can be worked on to make the system commercially viable. We expect tremendous saving on such cryptographic modules due to its intensive use of the resources.

6.2.1 EMI/EMC Testing

One of the requirements of FIPS certification is EMI/EMC compatibility. Once the final hardware/firmware is ready, the sensor nodes need to test for deployment in harsh environment. Such testing can be done by setting up the sensor node network in an environment with electro-magnetic interference with increasing magnitude. Based on the readings, the sensor node may have to undergo a design change or have an extra protective cover to achieve the targets required for the FIPS certification.

6.2.2 Tuning Code for Power Efficiency

The software currently developed for the Cryptographic Module MSP 430 is generic. Although the software is written in such a way that it minimizes power consumption, the code can be further optimized for minimum instruction execution and thus utilizing minimum power. The other approach could be calculating the maximum power that can be supplied by the harvester and tune the code for staying in limits of these power figures. Another way of reducing overall power consumption could be adjusting the time
frame of master message beats such that it utilizes minimum processing time of the Cryptographic Module.

### 6.2.3 Implementation on Impact-RLW sensor nodes

The code for the Cryptographic Module MSP430 has been developed on Texas Instruments dev kits for the MSP430. As a result, the programming has been done using JTAG interface and all the testing has been in-circuit emulation. Once fully developed, the code needs to be transferred on the sensor nodes and tested for full functionality. Also it will be advisable to check for power consumption on these boards also as the power figures might be different on the sensor nodes and the dev-kits.
Bibliography


A.1 NIST Approved Standard Algorithms for a Cryptographic Module

The below text has been taken from the NIST publication (National Institute of Standard and Technology, 2002). It is a list of the approved algorithms that can be implemented to achieve the required functions within the FIPS 140-2 standard.

**Symmetric Algorithms**: Advanced Encryption Standard

- Data Encryption Standard
- Triple Data Encryption Standard
- Escrowed Encryption Standard

**Asymmetric Algorithms**: Digital Signature Standard

- Revised DSS

**Hash Algorithms**: SHA-1

- SHA-224
- SHA-256
- SHA-384

**Random Number Generator Algorithms**: RNG for DSA

- RNG for RSA
- RNG for ECDSA

**Key Management Algorithms**: ANS X9.42

- AND X9.63
A.2 Data Elements

A.2.1 Heartbeat Message

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Type</td>
<td>8</td>
</tr>
<tr>
<td>Status</td>
<td>16</td>
</tr>
<tr>
<td>Battery Voltage</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 0-1: Heartbeat Message elements

A.2.2 Configuration Message

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Type</td>
<td>8</td>
</tr>
<tr>
<td>Device Class</td>
<td>16</td>
</tr>
<tr>
<td>Firmware Major Version</td>
<td>5</td>
</tr>
<tr>
<td>Firmware Minor Version</td>
<td>5</td>
</tr>
<tr>
<td>Firmware Release Number</td>
<td>6</td>
</tr>
<tr>
<td>S(^{\text{NAP}})™ InfoSensor Configuration</td>
<td>400</td>
</tr>
</tbody>
</table>

Table 0-2: Configuration Message elements

A.2.3 Firmware Status Message

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Type</td>
<td>8</td>
</tr>
<tr>
<td>UID</td>
<td>32</td>
</tr>
<tr>
<td>FW Packets: Total</td>
<td>16</td>
</tr>
<tr>
<td>FW Packets: Current</td>
<td>16</td>
</tr>
<tr>
<td>Image Number</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 0-3: Firmware Status Message Elements
A.2.4 Waveform Body Message

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Type</td>
<td>8</td>
</tr>
<tr>
<td>Resend</td>
<td>1</td>
</tr>
<tr>
<td>Waveform Sequence Number</td>
<td>15</td>
</tr>
<tr>
<td>Waveform Data</td>
<td>Variable</td>
</tr>
</tbody>
</table>

Table 0-4: Waveform Body Message
Appendix B

B.1 NIST Specified AES Algorithm

For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This is represented by $Nb = 4$, which reflects the number of 32-bit words (number of columns) in the State. For the AES algorithm, the length of the Cipher Key, $K$, is 128, 192, or 256 bits. The key length is represented by $Nk = 4$, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key (NIST AES). For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by $Nr$, where $Nr = 10$ when $Nk = 4$, $Nr = 12$ when $Nk = 6$, and $Nr = 14$ when $Nk = 8$ (NIST AES). The only Key-Block-Round combinations that conform to this standard are given in Table 8-1

<table>
<thead>
<tr>
<th>Key Length $Nk$ words</th>
<th>Block Size $Nb$ words</th>
<th>Number of Rounds $Nr$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>AES-192</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>AES-256</td>
<td>8</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 0-1: Key-Block-Round Combinations (NIST AES)

The Cipher is described in the pseudo code in Fig 8-1. The individual transformations -
SubBytes(), ShiftRows(), MixColumns(), and AddRoundKey() – process the State and are described in the following subsections.

---

**SubBytes( )**: The SubBytes() transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box).

---

Figure 0-2: SubBytes( ) function in AES (NIST AES)
**ShiftRows( )**: In the ShiftRows() transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, $r = 0$, is not shifted.

---

**MixColumns( )**: The MixColumns() transformation operates on the State column-by-column, treating each column as a four-term polynomial.

---

![ShiftRows Diagram](image1)

**Figure 0-3**: ShiftRows( ) function of the AES (NIST AES)

![MixColumns Diagram](image2)

**Figure 0-4**: MixColumns( ) function (NIST AES) of the AES
**AddRoundKey()**: In the AddRoundKey() transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of $Nb$ words from the key schedule.

---

![AddRoundKey Diagram](image_url)

*Figure 0-5: AddRoundKey( ) function (NIST AES) of the AES*