The Pennsylvania State University

The Graduate School

College of Engineering

POWER ANALYSIS OF CRYPTOGRAPHIC MODULE FOR WIRELESS SENSOR NODE

A Thesis in

Electrical Engineering

by

Ravikant Gupta

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

December 2010

The thesis of Ravikant Gupta was reviewed and approved* by the following:

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

Julio V. Urbina
Assistant Professor of Electrical Engineering

W. Kenneth Jenkins
Head of Department
Professor of Electrical Engineering

*Signatures are on file in the Graduate School

Abstract

Wireless sensor networks, which are collections of spatially distributed sensor nodes controlled and monitored by a master terminal, often must send sensitive data between master and node, which requires that the data be encrypted. Additionally, wireless sensor nodes are often self powered, thus node power consumption must be minimized.

This work presents a power analysis of a cryptographic module for wireless sensor networks. The core elements of the cryptographic module system are two Texas Instruments MSP430 microprocessors: one functioning as master, the other as slave. This cryptographic module is seeking to be certified to the Federal Information Processing Standard 140-2.

Master–slave communication is implemented through a Serial Peripheral Interface bus, which facilitates ciphering, deciphering, and transfer of sensitive data using the AES Advanced Encryption Algorithm. Implementation of FIPS-approved AES and Diffie–Hellman key exchange cryptographic algorithms was a system requirement. Authentication and attack prevention techniques include Known Answer Test, Error Correcting Code, and Zeroization.

The cryptographic module power is provided through vibration energy harvesting; therefore, power management is critical. Accordingly, the cryptographic module design implements power management via component selection, code optimization, and implementation of a Low Power Mode. The cryptographic module was developed using Texas Instruments USB-Debug-Interface and IAR Embedded Workbench. Subsequent to verification and certification, the cryptographic module will be operationally deployed aboard U.S. Navy ships.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Abbreviations

ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
AVG	Average
CSP	Critical Security Parameters
CLK	Clock
CISC	Complex Instruction Set Computing
CM	Cryptographic Module
CRC	Cyclic Redundancy Check
DES	Data Encryption Standard
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standards
FCS	Frame Check Sum
JTAG	Joint Test Action Group
KAT	Known Answer Test
LED	Light Emitting Diode
NIST	National Institute of Standards and Technology
PC	Personal Computer
PIN	Personal Identification Number
PIC	Programmable Interface Controller
RSA	Rivest, Shamir, and Adleman ciphering algorithm
RNG	Random Number Generator
RISC	Reduced Instruction Set Computing
RC4	Rivest Cipher 4
RC5	Rivest Cipher 5
RAM	Random Access Memory
SPI	Serial Peripheral Interface
TI	Texas Instruments
UID	Unique Identifier
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
ZIF	Zero Insertion Force

Symbols

µ	micro
@	at the rate of
+	addition
*	pointer
#	number

ACKNOWLEDGMENTS

I would like to thank first and foremost my adviser, Dr. Sven Bilén, for accepting me to work under him and for his support and guidance as my graduate advisor. He has provided me with great insight and feedback at every step. 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 am also thankful to Prof. Julio Urbina for providing me his valuable comments and feedback. I would also like to acknowledge Bob Capuro for helping me throughout my research. His timely inputs and suggestions have helped me enormously during my work. I am also thankful to Graduate office staff, especially SherryDawn for helping me with administrative paper work and meeting the deadlines.

I wish to thank KCF Technologies of State College, PA, which supported some of the work described in this thesis via a Navy STTR.

Last, but not the least, I would like to thank all my friends and family for their love and support during my research work at Penn State.

Chapter 1
Introduction

This chapter provides an overview of wireless sensor networks and sensor nodes used in such networks. This chapter also provides an introduction to the concepts of cryptography and its need in various sensor networks. A brief introduction is provided to the FIPS standard and the need for low power usage in cryptographic modules.

1.1 Wireless Sensor Network

A wireless sensor network (WSN) consists of various sensor nodes controlled by a central master terminal. These nodes are a combination of sensor technology, microprocessor, power source, and wireless communication interface as shown in Figure 1-1. WSNs are deployed using a particular topology in an area of interest to allow monitoring of environmental conditions and events. In such a wireless network, all nodes collect and forward data to one or several central controller nodes.

The main components of sensor node are: controller, transceiver, external memory, power source, and sensors, each described here in brief.

- Controller: Performs tasks, processes data, and controls the functionality of other components in the sensor node. The most commonly used controller is a microcontroller due to their low cost, flexibility to connect to other devices, ease of programming, and low power consumption; other options are microprocessors, FPGAs, and ASICs, which may be selected depending on the requirements of the WSN.
- Transceiver: The functionality of both transmitter and receiver are merged into a single device called a transceiver, which is used in a sensor node for communication. Sensor

nodes often make use of the ISM (industrial, scientific, and medical) bands as they provide unlicensed radio spectrum allocation and global availability. The various options for wireless transmission media are radio frequency, optical, and infrared. Radio frequency (RF) is generally the best fit for most WSN applications.

- External memory: Due to energy constraints, most commonly used memory devices are the on-chip memory in the microcontroller and flash memory. Flash is used due to its cost and large memory capacity.
- Power source: Power in a sensor node is required for sensing, communication, and data processing. Of these, data communications is the most power consuming operation. Commonly used sources of power are chargeable and non-chargeable batteries, but nowadays power-harvesting techniques enable nodes to be self-powered (as in our case). Sensors: These are devices that produce responses to changes in temperature, vibration, and other physical and environmental stimuli. A sensor generally provides a continuous analog signal that is then digitized by an analog-to-digital converter (ADC) and sent to the controller for further processing. The S^5NAP sensor node, which is a proprietary design by Impact RLW, Inc., was used in this work. More information and design details are provided in Chapters 3 and 4.

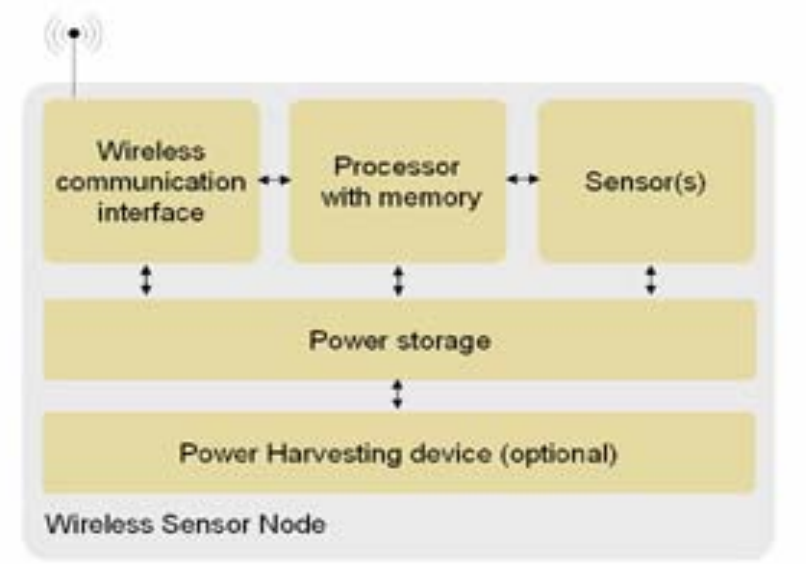


Figure 1-1: Architecture of sensor node [from coalesenses.com]

1.2 Cryptography

Cryptography is defined as the science of converting raw data into scrambled code that can be deciphered at other end. Cryptography uses two types of encryption: symmetric and asymmetric. Symmetric encryption uses the same key for encryption and decryption, whereas asymmetric uses different keys and, because of that, asymmetric is proven to be more secure.

In cryptography, a cipher is an algorithm that is used for performing encryption and decryption of raw data. The need for sending the data between two parties without the fear of getting intercepted by a third party has driven the evolution of ciphering. The ciphering procedure depends on a piece of information called a key that can be generated using different key-generating algorithms such as Diffie–Hellman key exchange, Elliptical Curve Cryptography (ECC), Digital Signature Algorithm (DSA), and RSA (Rivest, Shamir, and Adleman). Most ciphers can be categorized into two types: block cipher and stream cipher. In block cipher, operations are performed on a fixed length of bits/bytes called blocks; for example, a block

cipher will take set of 128 bits as input and, after encryption or decryption, it will return 128 bits of information. In a stream cipher, plaintext bits are combined using exclusion or operation with pseudorandom cipher bit stream called a key-stream.

When digital data is required to be securely stored in or exchanged between computers, then based on the sensitivity of the information, the need for protecting this data from unauthorized access must be considered. These requirements lead to a need for protecting this data which can be accomplished through ciphering.

The development of digital computers has made it possible to implement even the most complex ciphering algorithms available. As the complexity of the algorithm increases, so does the need for computational resources, which has lead to the development of modern-day cryptographic modules (CMs). A CM integrates cryptographic algorithms in software and hardware elements into a dedicated ciphering unit.

A CM first generates a key using a key generation algorithm and then takes a block or stream of raw data (depending on the type of algorithm used for encryption) and ciphers the data. The encrypted data can then be sent over a channel without worrying about interception by a third party. At the other end, there is a counterpart CM doing the exact opposite process called decryption. In this manner, data is securely transferred between different ends. Figure 1-2 shows the structure of a typical CM.

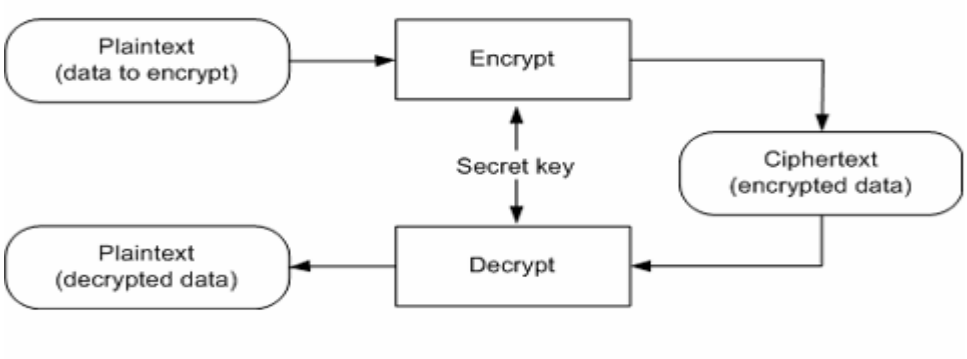


Figure 1-2: Block diagram of CM

CMs have been developed to secure important information, but poor design or weak algorithms can make the ciphering process insecure and place highly sensitive information at risk. Ideally, a CM should be implemented such that the ciphered data can only be deciphered by a module of the same type. This has lead to series of requirements and procedures and to the development of the Federal Information Processing Standard (FIPS).

1.3 FIPS Overview

With the requirement to use cryptography by the military and other federal agencies, a standard was needed to provide security assurance for the data transferred. In January 1994, the National Institute of Standards and Technology (NIST) released the Federal Information Processing Standard. The FIPS standard tests and validates a CM and its underlying algorithm against established standards to provide security assurance.

A CM that follows these specific rules as stated by NIST can seek what is known as FIPS certification. This certification updates its rules every few years, latest being FIPS 140-2 released in 2002. In FIPS 140-2, there are different levels of security certification, which depend on

which security conditions are satisfied. The higher the security level, the stricter are the security requirements.

1.4 Need for Low Power Consumption

With the increased use of embedded systems in industries such as consumer electronics, home automation, medical, security, and many others, the need for low power consumption has become important. This is because many embedded systems are battery powered and by choosing a system that consumes the lowest power, the battery life can be extended.

Low power consumption has always been a key concern for hardware developers, but there are limits based on the hardware selected. An active system can minimize power usage through proper hardware design and by optimizing the manner in which that hardware is used, i.e., by optimization with respect to power of the embedded software. This software has generally been written with considerations such as optimization of memory, but to minimize power usage and to achieve a system that consumes low power, optimization of the software with respect to power is also required.

There are a number of ways by which power can be optimized through software, such as using the low power mode(s) of the microcontroller, minimizing the current consumption by controlling processor frequency, minimizing the voltage level of the controller's supply, code optimization, among others. Each of these methods is explained in more detail in the following chapters.

1.5 System Overview

The sensor nodes for which the CMs have been developed in this work are called S^5NAP, which is a product of Impact RLW Systems, Inc., of State College, PA. The system consists of a

wireless sensor network that is capable of transferring the data between different nodes. The sensor nodes present in the system collect data of interest (e.g., temperature, vibrations, humidity, and other various measurements) and transfers this data to other nodes using a radio link. These sensor nodes are self powered as they generate their power from vibration-based energy-harvesting technology. Figure 1-3 shows a mesh sensor network connected to a host hub where all the information is collected and stored.

The sensor node contains a low power CM to securely transfer the data between different nodes using various FIPS approved algorithms. The CM consists of two Texas Instruments MSP430 processors, which act in master–slave configuration. The master MSP430 collects the data from the sensor nodes and the slave MSP430 ciphers this data. Henceforth, this will be referred to as the MSP430 crypto module.

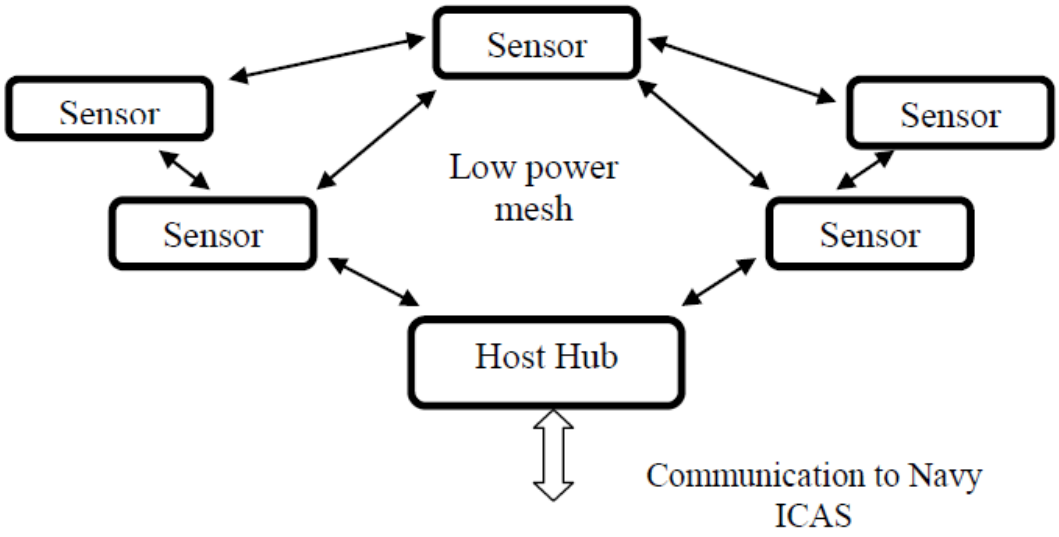


Figure 1-3 Sensor Node Mesh Network [KCF Technologies, 2008]

1.6 Contributions of this Work

Our primary contribution in this project has been to design a low power CM that is capable of encrypting and decrypting data using various FIPS approved algorithms. The software developed for the CM has been designed with low power usage as a key objective. This was mainly achieved by using low power modes of the microcontroller and optimizing the code used for encryption and decryption.

Furthermore, the module we developed can be used in almost any device that requires data security and low power consumption. The module can be used by any wireless senor network independent of any routing protocol used for communication.

1.7 Organization of this Thesis

The remainder of the thesis is organized as follows. Chapter 2 provides background information and a literature review. It also provides information about the CM and details of FIPS security requirements. Chapter 2 covers the selection process for the microcontroller and algorithm used for encryption and decryption. Chapter 3 provides information about the system architecture and the various protocols used for communication between different system elements. It also provides detailed descriptions of the roles of the system modules. Chapter 4 overviews the power analysis of the CM, the methods by which low power consumption can be achieved, and the workbench used to develop the module. Chapter 5 provides the testing results of the CM, specifically the results from the power analysis of the system. Finally, Chapter 6 concludes the work and suggests future work.

Chapter 2
Background

This chapter provides an overview of past work performed in this field and other similar cryptographic modules. Also covered is a study of microprocessors for low power usage and the selection process that chose the MSP430, which included factors such as key size, packet size, type of algorithm, and number of cipher rounds.

The second half of the chapter reviews in detail the Federal Information Processing Standard (FIPS) requirements for certifying a CM. As described earlier, NIST has specified a number of security requirements that must be satisfied by any CM used by a federal agency. This chapter also describes the need for FIPS and gives a background that led to the development of FIPS.

2.1 Background of Cryptographic Modules

The goal of this project was to design a system that can implement certain algorithms specified by FIPS and to minimize the power consumption of this system using various methods for power management.

2.1.1 Similar Modules

A number of CMs have been developed by other groups. One such device is the “Telos”, which is an ultra low power wireless sensor module for research and experimentation purposes developed by the University of California, Berkeley [*Polastre et al*., 2005]. Telos was built with three goals in mind: ease of use, minimum power consumption, and increased software and hardware robustness. It consists of a TI MSP430 microcontroller, Chipcon IEEE 802.15.4–complaint radio, and USB interface.

Another such device, released in 2001, is the “Mica” [*Hill et al.*, 2002], which was designed to serve as a general purpose platform for WSN research. Mica was useful for development, but unsuitable for deployments purposes because of its short range for radio communication. To overcome the shortcomings of Mica, “Mica2” was developed, which used an ATmega128 microcontroller and Chipcon CC1000 transceiver offering tunable frequencies from 300 to 900 MHz. To continue the Mica family, “MicaZ” was released in 2004 and uses a CC2420 radio, which is an IEEE 802.15.4–compatible radio.

A single-chip mote implementation called “Spec” [*Hill*, 2003] resulted from the Mica platform. Spec uses a number of dedicated hardware accelerators to perform encryption. Unlike the Mica family, Spec is fully integrated and offers limited interface flexibility.

2.1.2 Microprocessor Selection

To select the microprocessor used in this project, we performed a comparative study of a large subset of the available microprocessors on the market. For example, the PIC microprocessor class, particularly the PIC16F87, is very power efficient with features that include four low power modes and two-speed oscillators. However, after careful study of the available options, the Texas Instruments MSP430FXXX was selected due to its overall power consumption and its low power modes [*Kanani*, 2009]. For each of the processors considered, specifications as derived from data sheets are presented in Table 2-1.

The MSP430 was found to have the lowest power consumption in sleep and active modes. The microcontroller operates down to 1.8 V, which is important depending on the power source, e.g, AA batteries have a cutoff of voltage of 0.9 V and if two such batteries are used the system

cutoff voltage will be 1.8 V. Compare this to the Atmel processor, which can only run down to 2.7 V, leaving some of the power in the AA batteries unused.

Table 2-1: Power consumption for microprocessors evaluated [*Kanani*, 2009]

Company	**Processor**	**8 or 16 bit**	**# of Power Modes**	**Power Consumption**			
				Mode 1	Mode 2	Mode 3	Mode 4
Microchip	PIC16F87/88	8 bit	4	152 µW @ 1 MHz	14 µW @ 31.25 kHz	18 µW @ 32 kHz	0.2 µW
Atmel	AT89C5115	8 bit	3	3.7 mW @ 1 MHz	2.6 mW @ 1 MHz	NA	—
Texas Instruments	MSP430F1611	16 bit	3	726 µW @ 1 MHz	2.2 µW @ 1 MHz	0.2 µW	—
Maxim	MAXQ2000	16 bit	2	4.75 mW @ 14 MHz	12 mW @ 32.77 kHz	NA	—
EM Microelectronic	EM6812	8 bit	4	360 µW @ 1 MHz	6 µW @ 32 kHz	0.8 µW	0.16 µW

The MSP430 has the fastest wakeup time of all the microcontrollers evaluated, transitioning from standby mode to active mode in 6 µs. The MSP430 also has a DMA controller to reduce load from MCU core, which lowers the power consumption and increases the performance. In addition, the MSP430 has the advantage of having the largest on-chip buffer (10 kB), which is very useful for on-chip signal processing.

2.1.3 Power Consumption for Different Cryptographic Algorithms

Cryptographic algorithms are known to be computationally intensive as they consume a lot of resources such as memory space, CPU cycles, and power. Wireless devices are mostly battery operated, so power consumption due to running the cryptographic algorithm becomes a significant consideration. Hence, selecting the proper algorithm to meet cryptographic requirements is of utmost importance.

Energy consumption by cryptographic algorithms depends on various factors, such as the size of the block of data to be ciphered (assuming block ciphering), the size of the key used, and the number of cycles used in ciphering or deciphering the data. In the following sections, the effect of these factors is described in more detail.

2.1.3.1 Effect of Packet Size on Energy Consumption

Packet size is an important factor in wireless networks. Transmitting long packets improves the network utilization because there is overhead information that has to be transmitted with each packet. On the other hand, as the size of the packet increases, the error rate also increases [*Lettieri et al.*, 1999].

The packet size also has an effect on energy consumption as every encryption requires a key expansion process that consumes a constant amount of energy independent of the size of the key used. According to *Prasithsangree and Krishnamurthy* [2003], long packets consume less energy than short packets using same key length and number of operations. From work done by *Kiratiwintakorn* [2005], Figure 2-1 shows the energy consumption comparison between different algorithms as a function of packet size while keeping constant the key size and number of rounds. For this experiment, an 800-MHz, mobile Pentium III was used. As can be seen from the figure, AES consumes least amount of energy compared to the other methods when encrypting small packets, which is an important result for the current project as our system used 80-byte packets.

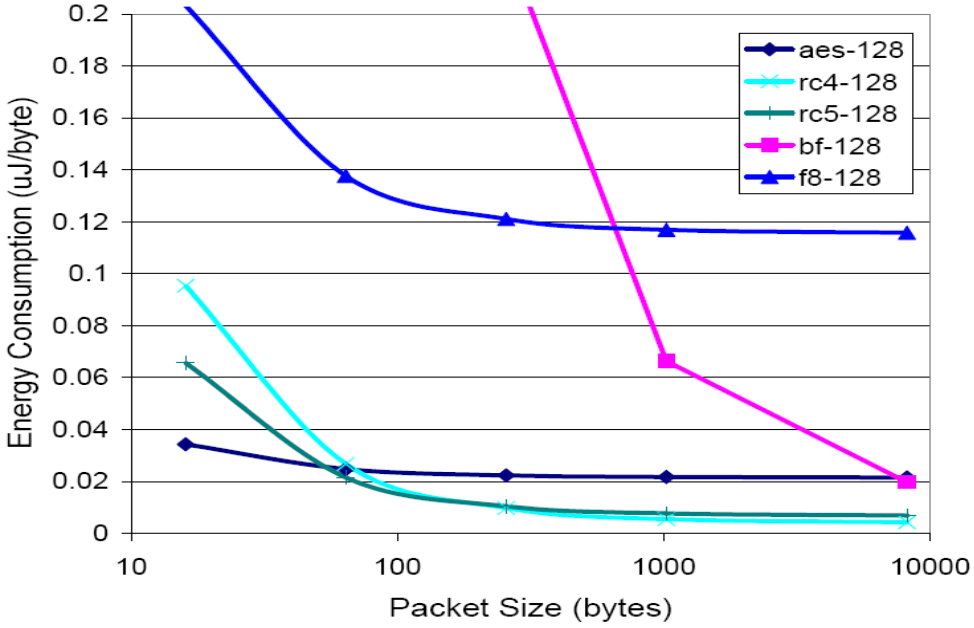


Figure 2-1: Energy consumption versus packet size using various algorithms [from *Kiratiwintakorn*, 2005]

2.1.3.2 Effect of Key Size on Energy Consumption

We now discuss the effect of key size on energy consumption of a cipher algorithm. We consider two cases: with and without key expansion. The difference between these two is that, for the case without key expansion, the number of cycles is counted after the key expansion process. Figures 2-2 and 2-3 show the amount of energy consumed per byte using different key sizes, packet sizes, and algorithms with and without key expansion. As shown in the figures, the amount of energy consumed in the case of AES depends on the size of the key used, whereas all other algorithms (i.e., Blowfish, RC4, and RC5) are almost independent of size of the key in both with and without key expansion. This behavior of AES is due to the fact that, as the size of the key increases, the number of ciphering rounds increases, which means more CPU cycles and, hence, higher energy consumption. In the case of other algorithms, as the size of the key increases there is only a slight increase in the number of rounds of operation and, hence, no significant effect on energy consumption.

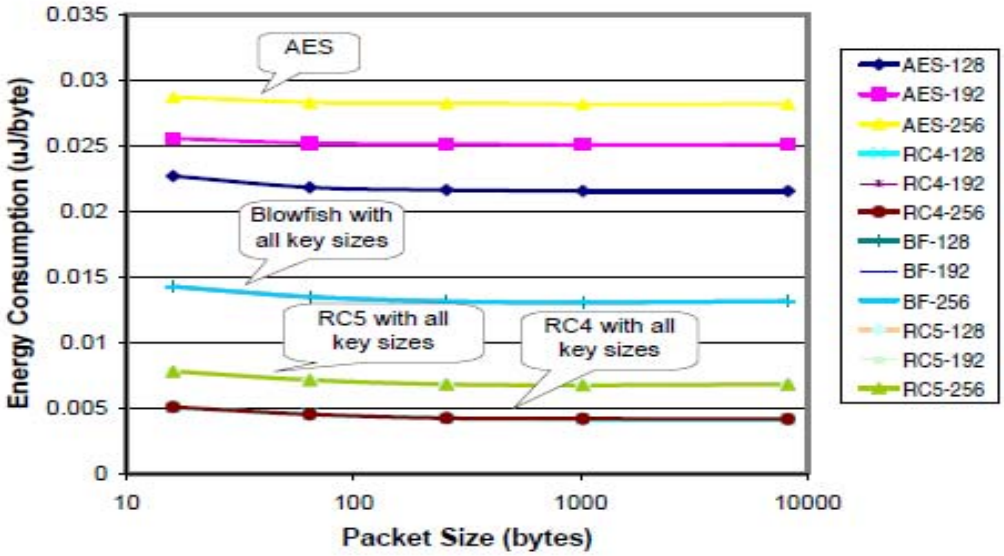


Figure 2-2: Energy consumption versus key size for various algorithms without key expansion [from *Kiratiwintakorn*, 2005]

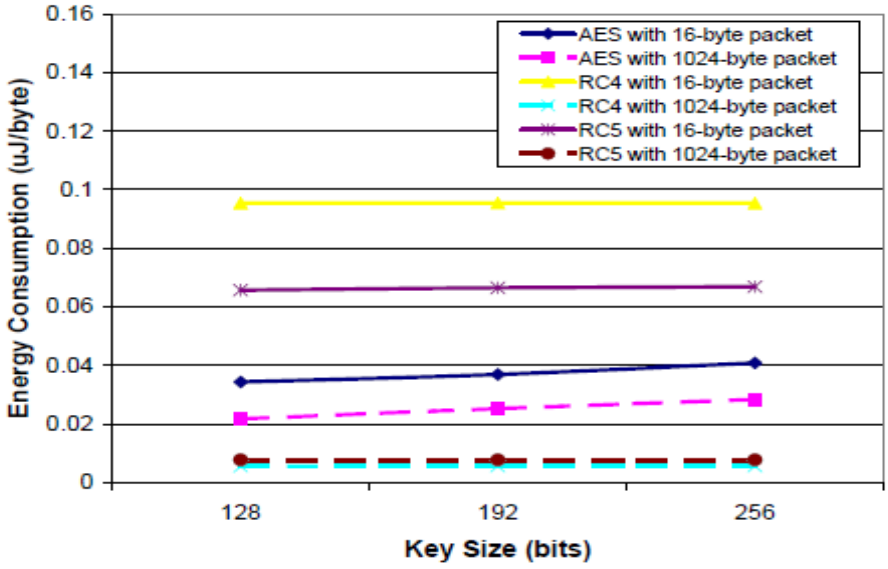


Figure 2-3: Energy consumption versus key size for various algorithms with key expansion [from *Kiratiwintakorn*, 2005]

2.1.3.3 Effect of Number of Rounds on Energy Consumption

The energy consumption in a cryptographic algorithm depends heavily on the number of ciphering rounds, especially in the case of block ciphers like AES and RC4. In a block cipher algorithm, the input data go through a number of repetitive rounds and, as the number of rounds increases, energy consumption by the algorithm increases. In the previous section, we saw the effect of key size on the power consumption of the AES algorithm, but the effect due to the number of operational rounds is much larger than for key size. According to the AES standard, there is certain number of rounds that must be performed depending on the key size. Key sizes of 128, 196, and 256 bits require 10, 12, and 14 rounds of operation, respectively. The standard requires particular rounds of operation because of the security considerations. The algorithm becomes more and more susceptible to cryptanalysis attacks as number of rounds of operation decreases. Figure 2-4 shows the effect of number of rounds on energy consumption per byte for different algorithms.

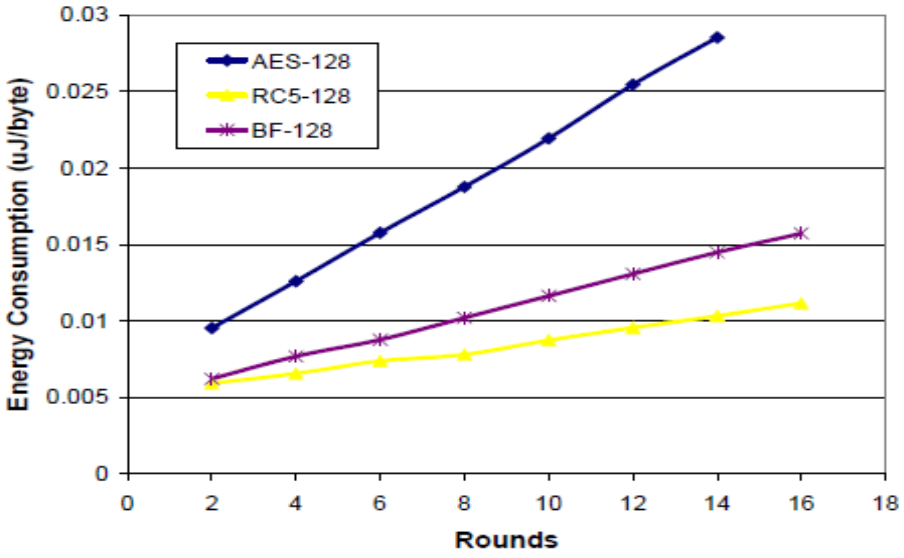


Figure 2-4: Energy consumption versus rounds of operation [from *Kiratiwintakorn*, 2005]

2.1.4 Power Consumption for Key Generation Algorithms

Cryptographic algorithms require a key generation algorithm so that the key can be generated from time to time and shared securely between different nodes of the WSN. There are various public key-generation algorithms available, such as Diffie–Hellman Key Exchange, Elliptical Curve Cryptography (ECC), Digital Signature Algorithm (DSA), and RSA (Rivest, Shamir, and Adleman). These algorithms consume different amounts of energy based on their complexity. In Figure 2-5 where the plot intersects the Y axis denotes the amount of energy consumed by different algorithms for key generation, while the slope indicate the energy needed to transfer data..

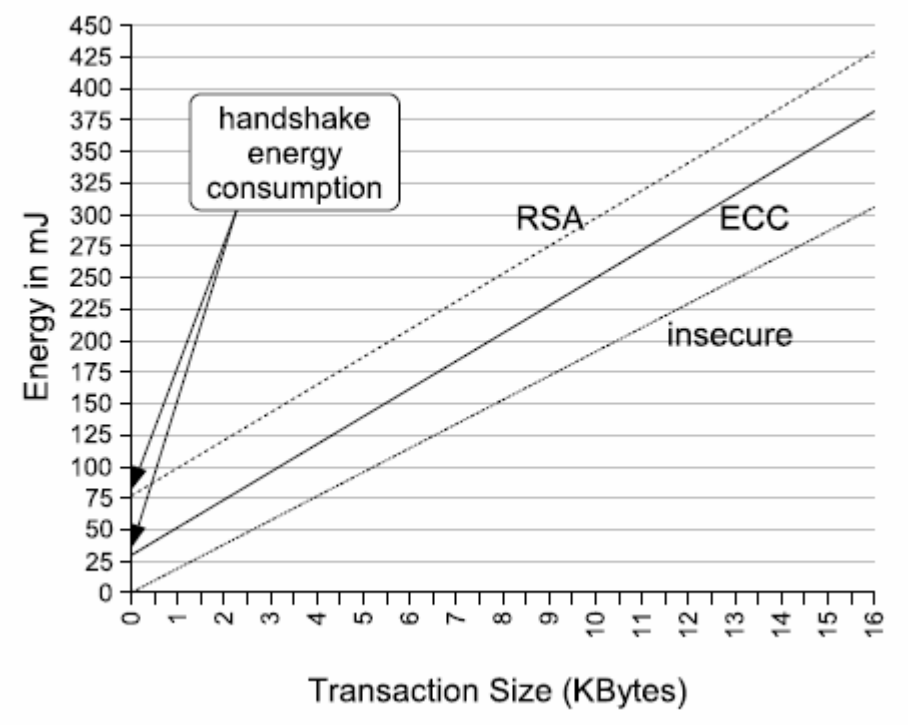


Figure 2-5: Energy consumption as a function of application data size [from *Gupta and Wurn*, 2008]

2.2 The Federal Information Processing Standard

As mentioned earlier, every CM used by a U.S. federal agency meet certain requirements as defined by NIST. A CM meeting these requirements, once certified, is awarded FIPS Certification (the current version is Version 140-2).

2.2.1 FIPS 140-2 Certification Security Levels

Since security requirements vary for different applications, FIPS provides four levels of security. It is incumbent on an organization to determine their requirement for level of security, which will depend on the sensitivity of the data and possible impact of interception. Security levels provided by FIPS vary depending on the requirements levied. Level 1 is the easiest to achieve and level 4 is the toughest.

2.2.1.1 Security Level 1

Security level 1 is the lowest level of security among the four levels provided by FIPS. Its does not require any particular physical security requirements besides the basic requirement for production grade component [*NIST,* 2009]. It allows software and firmware components of a CM to be implemented on any general computing system. This security level is best suited for low cost CMs for which physical and network security are limited or not required. For example, a personal computer encryption board will qualify for security level 1.

2.2.1.2 Security Level 2

Security level 2 provides better physical security compared to level 1. In this level, there is a requirement for tamper evidence, such as tamper-evident coatings or seals or pick-resistant locks. These coatings are placed such that it has to broken to get a physical access to the CM [*NIST,* 2009]. Security level 2 also requires a minimum level of authentication of the operator for

specified roles and services. Security level 2 imposes much stricter requirements on the type of operating system used by a CM. The operating system has to meet specific requirements as outlined in Annex B of the *NIST* [2009] document.

2.2.1.3 Security Level 3

Security level 3 has much stricter physical security requirements including detecting and responding to tampering. Level 3 prevents intruders from access to Critical Security Parameters (CSP). The physical security in level 3 may include tamper evidence and response circuitry for any physical tampering with the system.

Security level 3 has identity-based authentication requirements, which improves the security provided over that of level 2 [*NIST*, 2009]. Level 3 requires the CM to make sure that it is capable of authenticating the identity of the operator and verifies that the operator is authorized for the specific role or task.

2.2.1.4 Security Level 4

Security level 4 is the highest level of security defined by FIPS. At this level, the physical security mechanism provides a complete shield from any unauthorized attempts to gain physical access to the CM. Any penetration of the CM via any method will result in complete zeroization of all the CSPs in the module. This kind of security is most useful for unprotected environmental conditions [*NIST*, 2009].

Security level 4 also protects the module from any harsh environmental conditions that are outside of the normal operating ranges for voltage and temperature. A CM is required to detect any fluctuation in these values that occurs and to zeroize all CSPs when out of range.

2.2.2 Security Requirements

There are various design and implementation requirements for CMs that are needed to achieve FIPS certification. This include requirements on module port and interfaces; roles and services; finite state module; physical security; operational environment; and more.

2.2.2.1 Cryptographic Module Specification

A CM is a set or combination of software, hardware, and firmware that is capable of implementing cryptographic algorithms or processes, and optionally, a key-generation algorithm [*NIST*, 2009]. A CM should implement at least one approved security function used in approved mode of operation.

The "cryptographic boundary" defines the perimeter or physical boundary of the CM. Within that boundary should exist any software or firmware used in cryptographic functions and any hardware to support that firmware or software. When seeking certification, the application package should

- specify any hardware, software, or firmware that is not a part of these security requirements and explain the reason for the same;
- document and specify all input and output data ports of the CM;
- account for all logical control and logical status indicators of the CM;
- document all security functions approved and unapproved that are employed by the CM;
- document and specify a block diagram depicting all hardware components and the interconnections between different hardware elements of the CM; and
- document all security related information, such as public and private keys, authentication data, and CSPs.

2.2.2.2 Cryptographic Module Ports and Interfaces

A CM should restrict all data flow and access points to physical ports and logical interfaces that are defined as entry and exit points of the module [*NIST*, 2009]. A CM should have the following logical interfaces:

- Data Input Interface: all data including plaintext data, ciphered data, cryptographic keys, and CSPs that are processed by the CM shall enter through data input interface.
- Data Output Interface: all data including plaintext data, ciphered data, cryptographic keys, and CSPs, but excluding status data, shall exit through data output interface.
- Control Input Interface: all input commands, signals, and control data including the manual control like switches and keyboards shall enter through control input interface.
- Status Output Interface: all status data, indicators, and output signals used to indicate the status of the CM shall exit through the status output interface. These data may include return codes and physical indicators such as LEDs and displays.

All power entering the CM, including the external power from a power supply or battery, should enter through the power port. There is no requirement for a power port if the power is supplied internally to CM within its cryptographic boundary.

Each security level levies different requirements: for levels 1 and 2, physical ports and logical interfaces used for authentication data, CSPs, and cryptographic keys can be shared with other ports and interfaces of the CM; whereas, levels 3 and 4 require either physical or logical separation from other port and interfaces.

2.2.2.3 Roles, Services, and Authentication

A CM should be able to support authentication for operators and their respective roles, which should be implemented via software using a password mechanism.

2.2.2.3.1 Roles

A CM should be able to support the following roles for an operator:

- User Role: to perform general purpose security activities, which include cryptographic operations and approved security functions.
- Crypto Officer Role: to perform basic cryptographic initialization and management functions like input and output of cryptographic keys and module initialization.
- Maintenance Role: to perform physical or logical maintenance services. All the CSPs and plaintext keys should be zeroized when entering or exiting this role.

2.2.2.3.2 Services

The document describing the CM should provide all the services and functions that can be executed by CM, which should be able to provide the following functions:

- Show Status: output the current status of the CM.
- Perform Self Tests: initiate and execute the self tests.
- Perform Approved Security Functions: the CM should be able to perform at least one approved security function in Approved mode of operation.

The document for services should provide:

- both approved and non approved cryptographic functions and services;
- inputs, expected outputs, and authorized roles in which the service can be performed for every specific service provided by the CM; and

- any services provided by CM for which operator is not required to gain any Authorized role.

2.2.2.3.3 Operator Authentication

Authentication mechanisms may be required in a CM to authenticate the operator for doing some roles and functions and for verifying that the operator is authorized to do so. Depending on the security level, a CM should be able to support one of the following mechanisms:

- Role-based Authentication: in role-based authentication, the module shall require that one or more roles either 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: in identity-based authentication, the module shall require that one or more roles either be implicitly or explicitly selected by the operator and shall authenticate the identity of the operator and authorization of the operator to assume the selected role. If the CM allows an operator to change roles, then the module should check the authorization of identified operator.

2.2.2.4 Finite State Model

The functions of a CM should be specified using a finite state model represented by a state transition diagram or state table. The transition table should consist of:

- all error and functional states of the CM;
- corresponding transitions from one to another state;
- events that cause the transition; and
- output event resulting from the transition.

A CM should include following states:

- Power On/Off States: states for primary, secondary, or backup power.
- Crypto Officer States: states in which crypto officer services are performed.
- Key/CSP Entry States: states in which cryptographic keys and CSPs are entered in the CM.
- User States: states in which authorized user obtain security services and perform approved and unapproved functions.
- Self-test States: states in which the CM performs self tests.
- Error States: states in which the CM faces an error.

A CM may or may not contain other states such as Bypass and Maintenance.

2.2.2.5 Physical Security

A CM shall employ physical security mechanisms in order to avoid any unauthorized physical access to the contents of the module. All the important components including the software, hardware, and firmware should be protected. Table 2-2 below summarizes the physical requirements for each of the four security levels.

Table 2-2: Summary of physical requirements for each security level [NIST, 2009]

	General Requirements for all Embodiments	Single-Chip Cryptographic Modules	Multiple-Chip Embedded Cryptographic Modules	Multiple-Chip Standalone Cryptographic Modules
Security Level 1	Production-grade components (with standard passivation).	No additional requirements.	If applicable, production-grade enclosure or removable cover.	Production-grade enclosure.
Security Level 2	Evidence of tampering (e.g., cover, enclosure, or seal).	Opaque tamper-evident coating on chip or enclosure.	Opaque tamper-evident encapsulating material or enclosure with tamper-evident seals or pick-resistant locks for doors or removable covers.	Opaque enclosure with tamper-evident seals or pick-resistant locks for doors or removable covers.
Security Level 3	Automatic zeroization when accessing the maintenance access interface. Tamper response and zeroization circuitry. Protected vents.	Hard opaque tamper-evident coating on chip or strong removal-resistant and penetration resistant enclosure.	Hard opaque potting material encapsulation of multiple chip circuitry embodiment or applicable Multiple-Chip Standalone Security Level 3 requirements.	Hard opaque potting material encapsulation of multiple chip circuitry embodiment or strong enclosure with removal/penetration attempts causing serious damage.
Security Level 4	EFP or EFT for temperature and voltage.	Hard opaque removal-resistant coating on chip.	Tamper detection envelope with tamper response and zeroization circuitry.	Tamper detection/ response envelope with tamper response and zeroization circuitry.

2.2.2.5.1 Environmental Failure Protection

All electronic devices and circuitry are designed to operate within some particular environmental conditions; any deviation from the normal operating ranges of voltage and temperature can cause failure of electronic circuitry that can compromise the security of a CM [*NIST*, 2009]. Proper assurance that security of the CM cannot be compromised can be achieved by having the module employ environmental failure protection (EFP) features or undergo environmental failure testing (EFT).

2.2.2.6 Cryptographic Key Management

FIPS has some requirements for cryptographic key management for the entire life cycle of cryptographic keys, cryptographic key components, and CSPs. Key management mainly includes Random Number Generation (RNG), key generation, key establishment, key distribution, and key zeroization.

2.2.2.6.1 Random Number Generators

A CM may use an approved or an unapproved random number generator (RNG) in an approved mode of operation; however, the data output from RNG should pass the continuous random number generator test. The output from an unapproved RNG can be used for the following:

- as input to an approved deterministic RNG; and
- to generate or initialize vectors for approved security functions.

2.2.2.6.2 Key Generation

A CM can generate a key internally. If a CM generates keys for an approved algorithm or security function, then it should be generated using an approved key generation method. If an approved key generation method requires input data from the RNG, then an approved RNG should be used.

2.2.2.6.3 Key Establishment

There are various methods of key establishment, such as a manually transported key loading device, an automated public key algorithm, or a combination of both manual and automated. If key establishment methods are used in a CM, only approved methods should be used.

2.2.2.6.4 Key Entry and Output

Cryptographic keys can be entered into or output from a CM. If cryptographic keys are entered into or output from CM, the entry or the output of keys should be performed using either manual methods such as via a keyboard or electronic methods such as via PC cards or smart cards**.**

2.2.2.6.5 Key Storage

Cryptographic keys in a CM can be stored either in plain text or encrypted form. Plaintext and private keys should not be accessible from outside the CM to any unauthorized operator.

2.2.2.6.6 Key Zeroization

All CMs should provide methods to zeroize all plaintext and private keys within a module. This function is used mainly when there is a breach in physical security of the module by an unauthorized operator.

2.2.2.7 Self Tests

For ensuring that the CM is functioning properly, Power-up Self Tests and Conditional Self Tests should be performed. A Power-up Self Test, as the name suggests, should be performed when a CM is powered up and Conditional Self Test should be performed when an applicable security function or operation is invoked.

Chapter 3
System Architecture and Interface

This chapter provides an overview of the system architecture and the interface between different modules. The chapter also includes information on the flow of data from sensor nodes to main master terminal and back to the nodes. The entire process includes encryption, decryption, and key generation using various FIPS approved algorithms.

3.1 System Overview

Figure 3-1 provides a system overview for the wireless sensor network and highlights the location of the CM. In short, the system concept of operations can be described in four steps: first, the Master collects data from sensor nodes and send it to Crypto module for encryption/decryption. Then, after performing the particular function the Crypto module sends the data back to Master MSP430. The Master collects this ciphered data and sends it to the Dust Radio, which sends the packet to the required destination via the Dust Network.

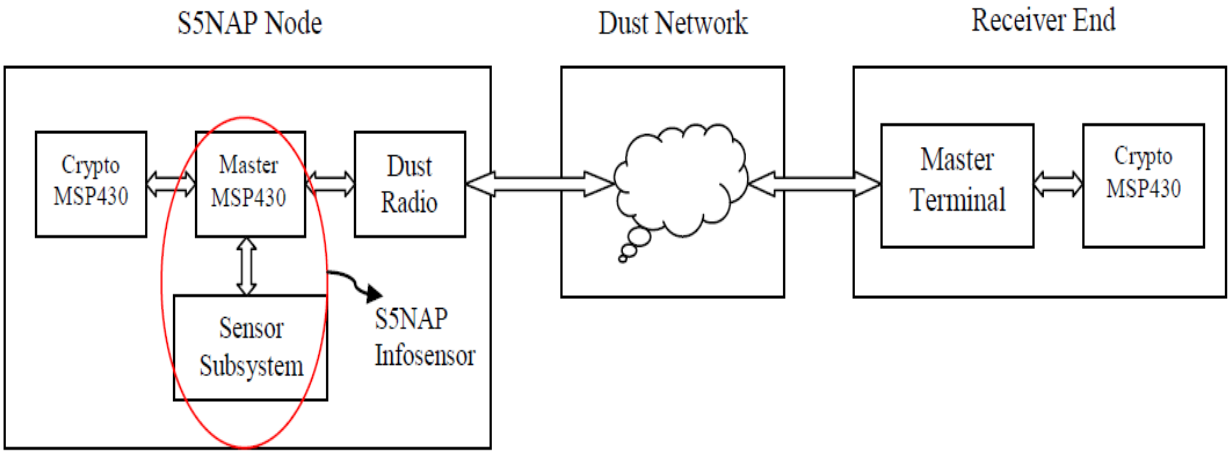


Figure 3-1: Wireless sensor network system overview

The WSN system architecture consists of three main subsystems: S^{5}NAP Node, Dust Network, and Receiver. The S^{5}NAP node consists of four major components: Master MSP430, FIPS MSP430, Sensor subsystem, and Dust Radio as shown in the Figure 3-1. The Master acts as the backbone of the system as all other components communicate through the Master. The receiver-side Master Terminal, which is mainly a connector program, is connected to the Dust Network on one side and the CM on the other side.

3.1.1 Master MSP430

The Master is one of the main components of the S^{5}NAP module, which carries out several important functions such as maintaining Serial Peripheral Interface (SPI) communication with the CM, providing a clock to the CM, and waking up the CM from sleep mode. The Master is also responsible for running the S^{5}NAP firmware, which in turn is responsible for collecting the sensor data from the ADC, reporting its status; transferring its data wirelessly using the Dust radio, and performing all the wireless firmware updates.

3.1.2 FIPS MSP430

The CM is responsible for encrypting the raw data supplied by Master so as to enable secure communication of this data over the air. It is also responsible for decrypting this encrypted data at the receiver end. The CM uses the Advanced Encryption Standard (AES) algorithm for encryption and decryption of data. After encryption or decryption, the output data are sent back to the Master terminal. Another important function of CM is to generate keys using the Diffie–Hellman key exchange algorithm. This key is used by the AES algorithm for encryption and decryption of data. The CM communicates only with the Master using identifier bits to let the Master know what kind of information is in the packet it is being sent.

3.1.3 Sensor Subsystem

The sensor subsystem is responsible for collecting the data of interest. In our case, this data are vibrations that are measured using an integrated charge-mode accelerometer. The output of the charge-mode accelerometer serves as the input to two circuits: a) the waveform acquisition circuitry and b) the continuous vibration monitoring circuitry.

3.1.4 Dust Radio DN 2140

The Dust DN2140 radio is a proprietary IEEE 802.15.4 wireless mesh networking solution provided by Dust Networks. The DN2140's communication protocol is proprietary in nature so it is like a black box in the S^5NAP system.

3.2 System Interface

There are several interfaces in the S^5NAP system, some software and some hardware. One of the most important interfaces is that between Master and the CM using the SPI bus. Other interfaces include that between Master and Dust Radio using the serial command interface and the interface between the Dust Radio and Dust Network via the IEEE 802.15.4 protocol.

3.2.1 Cryptographic Module to Master MSP430 Interface

The interface between Master and the CM is maintained using the SPI bus, which is the most important interface to the CM as most of the traffic flow takes place in this part of the module.

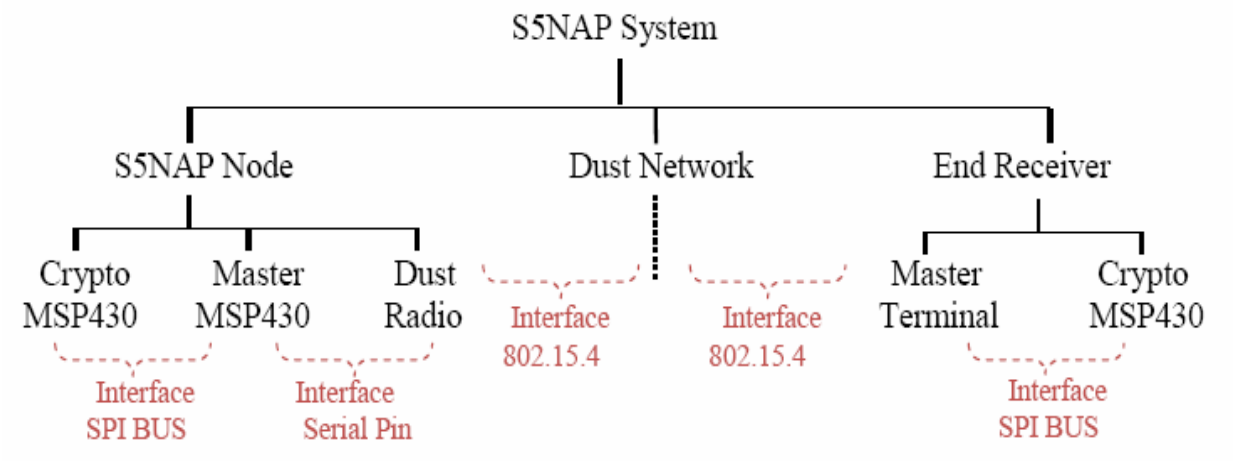


Figure 3-2: Overview of System Interfaces

The SPI bus is a synchronous serial standard that allows a master device to communicate with a slave device. SPI operates in full duplex mode. Devices communicate using a master–slave relationship, in which master generates a clock, selects a slave device, starts transmitting packets, and simultaneously receives packets as SPI communication is always in both directions. There is no error check in SPI, which requires that the master and slave check that the data received are understandable. Multiple slave devices can be used with individual chip select lines and, as such, SPI is called as "4-wire" SPI. The SPI bus specifies four logic signals (Figure 3-3):

- SCLK — Serial Clock from master;
- MOSI — Master Output, Slave Input;
- MISO — Master Input, Slave Output; and
- SS — Slave Select, active low from master;

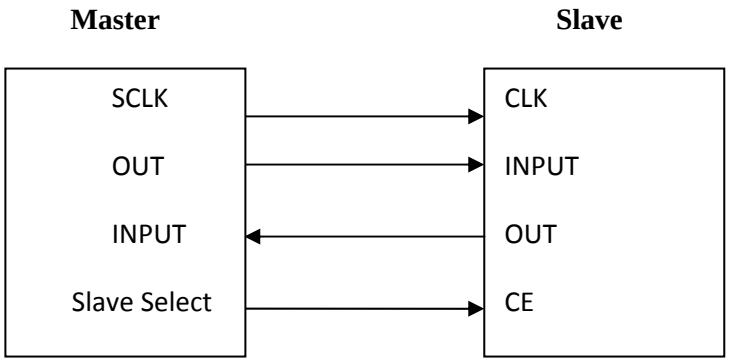


Figure 3-3: Basic interface between master and slave module

For our system, the complete pin interface is shown below in Figure 3-4. SPI _CLK is used by Master to provide clock to the Crypto MSP430; this clock plays a very important role for successful and error free communication between the two devices. Clock frequency can be varied to the processor depending on its function. In our system, we use a clock frequency of 4 MHz. The second connection is SPI_MOSI (SPI Master Out Slave In), which is the output line of Master MSP430. Similarly, SPI_MISO (SPI Master In and Slave Out) is the output line of the slave MSP430, i.e., the CM. FIPS_MCU_SEL is used by the master to select the CM, as the Master is connected to several slave devices. FIPS_MCU_INT is an interrupt line used by the Master to wake up the CM from the Low Power Mode (LPM). After waking up the CM, the Master starts sending and receiving bytes from CM, which, after processing the data (encryption, decryption, or key generation), generates the MAIN_MCU_INT interrupt and, as soon as the Master is ready, it again starts receiving the data sent by the CM.

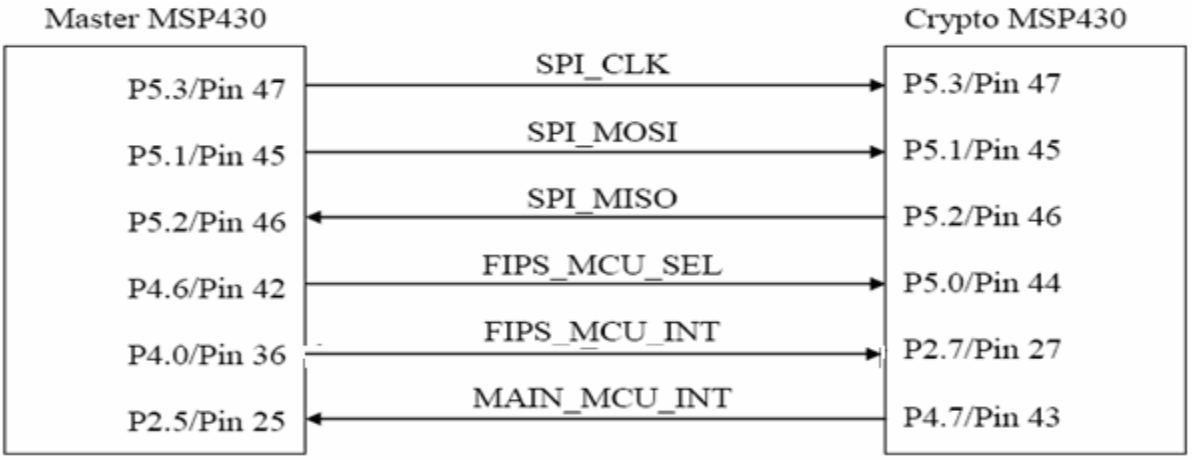


Figure 3-4: Pin interface between Master MSP430 and Crypto MSP430

The interface between the Master and CM is the backbone of entire logical system. The Master collects the vibration data from Sensor subsystem, which needs to be transmitted over the Dust Network. If this critical data are not encrypted, they could be intercepted and read by anyone over the air. So, this data has to be encrypted before transmission, which is accomplished by the CM. As soon as the Master receives data from Sensor subsystem, it wakes up CM and starts sending packets for encryption. The CM encrypts the data using the AES algorithm and sends it back to the Master. At the receiver end, the reverse process occurs, i.e., the Master sends encrypted packets to the CM for decryption, which then decrypts the packet and sends it back to Master.

For the CM to perform encryption and decryption of data using the AES algorithm, it must have a key for doing so. If the CM does not have the key, then it starts the key exchange using the Diffie–Hellman algorithm. Only after successful generation and exchange of the key is the CM capable of encrypting and/or decrypting the data.

3.2.2 Master to Dust Radio

The Master and Dust Radio communicate using a Universal Asynchronous Receiver/Transmitter (UART) at a baud rate of 9600. The Master sends the Dust Radio packets that contain ciphered data, a heartbeat message, and an identifier field. All the packets received are checked using 16-bit frame checksum (FCS) and the packets containing an FCS error are discarded. There is no mechanism by which the Master can be informed of a discarded packet. The Dust Radio receives packets from the network and forwards them to the Master. Along with packets Dust Radio also attaches the originating address of the packet to the Master as this is important information for Master when it is passing the packet to CM. CM use this information to know where the packet has come from. Similarly when the packet passes through the Dust Radio into the Network, the Dust Radio appends a header that contains important information regarding the network, such as the routing path, number of hops, destination address, etc. So, when the packet is received at the other side, the Dust Radio removes the header and passes the rest of message to the Master.

Another important thing to note is that when the packet is sent from CM it contains information about the destination address. But, this part is clipped off when it forwarded to the Dust Radio to save some bytes.

Chapter 4
System Implementation

This chapter is divided in to three parts. The first part describes the CM algorithm and the functions performed by the CM. Next, the flow of packets in the WSN system from one node to another is described. A new system specific algorithm was designed is explained. Finally, this chapter gives an overview of the test bench used to implement the system and also shows some test inputs and corresponding outputs of the system.

4.1 Cryptographic Module

As mentioned above, the CM performs a series of operations and algorithms on the data received from Master. The CM goes through several stages when performing these operations; these stages are explained below in detail and flow charts for stage transitions are provided. To maintain similarity between the two ends, there is a similar CM at the connector end to implement the same functions.

4.1.1 Cryptographic Module States

The CM goes through various states when performing encryption/decryption and key generation. Figure 4-1 shows these states in the order of their occurrence. Brief descriptions of all the states of the CM are given below:

- Initialization State: In this state the CM initializes various buffers it is going to use, such as its buffers for storing keys and input and output data. It also initializes various pointers to these buffers and functions for encryption, decryption, key generation, storing and stuffing keys, and the RNG function. In this state the CM also starts preparing the

communication channel with the Master. In order to do this, it initializes the SPI bus and initializes the interrupt lines to the Master.

- Sleep State: After preparing the channel for communication and initializing the variables, CM goes into a sleep state (i.e., LPM4). This state is very important as power consumption in this state is very low, which helps in power conservation for the module.

- Active State: As soon as the Master gets data from sensor node it interrupts the CM. The module goes back into active mode and waits for a clock signal from the Master.

- Tx and Rx State: The CM waits for a Tx interrupt signal from the Master and, as soon it sees that interrupt, there is a exchange of 80 bytes from the Master to the CM and vice versa. The data sent by the CM to the Master, however, contains garbage values. The reason for this is that SPI Communication is a two-way communication link, so a module has to transmit and receive at the same time. When the CM is done processing the data, it sends the Master an interrupt signal and again there is an exchange of 80 bytes.

- Data Processing: In this state the CM process the data it received and perform functions such as encryption, decryption, and key generation. Again, as mentioned earlier, the type of operation that the CM performs depends on the packet identifier bits.

Figure 4-1 shows all the states mentioned above and the transitions between them. The CM starts from the Initialization state and, after initializing the variables, it goes to the Ready state in which initialization of the SPI bus takes place. After the Ready state, the CM sits in Sleep mode until it is interrupted by the Master interrupt (FIPS_MCU_INT), which brings the CM to the Active state. In the Active state, the CM waits for a clock from the Master and then goes back to the Sleep state until it is again interrupted by the Master interrupt (Tx interrupt) signal. As soon

as the CM gets the Tx interrupt, it goes into the Tx and Rx state, in which it exchanges 80 bytes with the Master. After the data exchange, the CM goes into the Data Processing state, in which it processes all the data depending on the identifier bits. After processing the data, the CM goes back into the Tx and Rx state, which it sends the processed data to the Master. This completes one cycle of the CM, after which it goes back into Sleep mode.

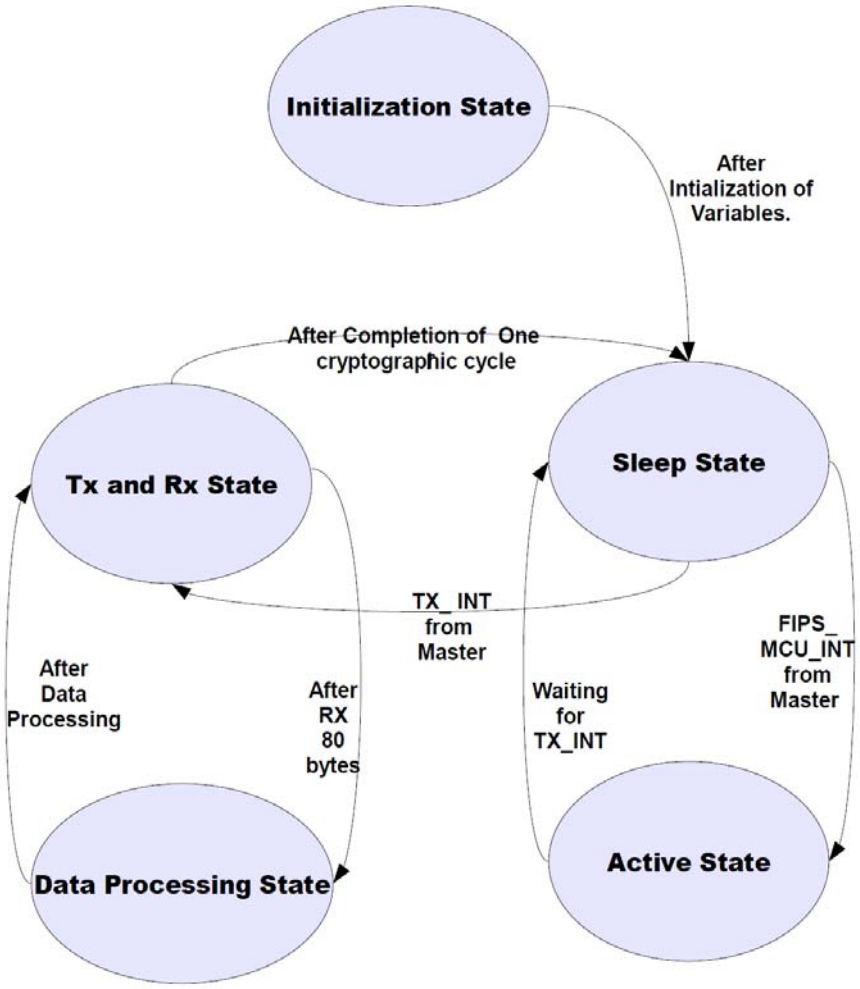


Figure 4-1: State diagram of the CM

4.1.2 Cryptographic Functions

Detailed descriptions for all the operations and functions performed by CM are provided below.

4.1.2.1 Encryption/Decryption

After careful examination of all the algorithms available, AES was chosen as the encryption algorithm for this project mainly because of its low energy consumption when used for small packets. The basic steps of the AES algorithm are described briefly below:

- Substitute Bytes: this is a nonlinear substitution and one of the main reasons for the security of the AES algorithm. This step can be considered as lookup table step. Using this lookup table, the 16 bytes are substituted by the respective values found in the table.
- Shift Rows: as the name implies, this step processes different rows. A simple rotate with different rotation width is performed. For example, if we have 4×4 bytes of input data, the second row will be shifted one byte to the left in the array, the third by two bytes, and the fourth by three byte positions. The first row is not shifted.
- Mix Columns: this is one of the most complex operations to implement in software. The Mix Column operation is pretty similar to “Shift Row”, with the only difference being it works on columns instead of rows. To make this operation reversible at the other end, instead of multiplication and addition, Galois field operations are used.
- Add Round Key: is a very simple step in the entire AES algorithm. The corresponding bytes of the input data and the expanded key are XORed.
- Key Expansion: This is the last step in the AES algorithm. Considering the case of 128-bit AES, in this process 128 bits of the original key are expanded into eleven 128-bit round keys.

Figure 4-2 below shows the implementation of AES:

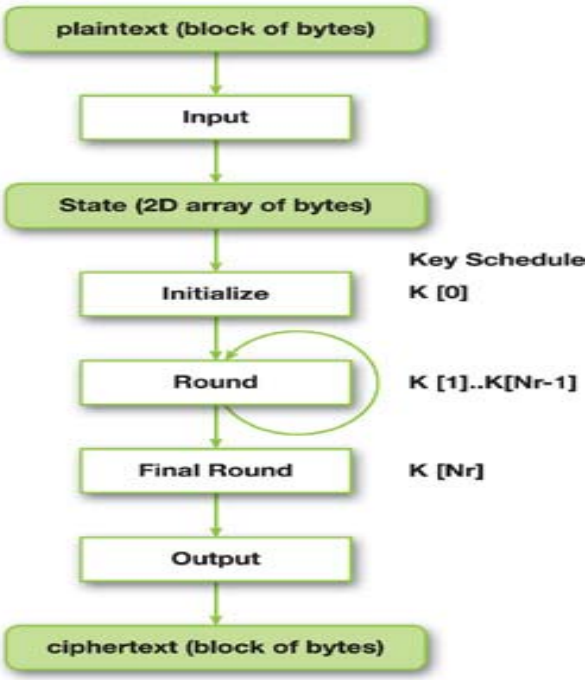


Figure 4-2: Illustration of AES algorithm [from Nvidia.com]

4.1.2.2 Key Generation

As mentioned before, the Diffie–Hellman key exchange algorithm is used for key generation in this project. The Diffie–Hellman key exchange method allows two different nodes, with no previous knowledge about each other, to jointly establish a shared secret key over an insecure channel. This shared secret key can be of variable length, which determines the strength of the ciphering.

With reference to Figure 4-3, we describe the Diffie–Hellman key exchange method below. In this method, there is a prime number P and a generator g that are known to all the nodes in the WSN. Now, suppose that Alice and Bob want to generate a shared symmetric key, they already know the value of P and g. Alice generates a random number a and Bob generates a random number b. Then each of them computes their public value using $g \times \text{mod } P$:

Alice's public key : $g\ a \bmod P$

Bob's public key: $g\ b \bmod P$

Now these values are exchanged and a final private key is generated at each node:

Alice computes $g\ a\ b \bmod P = (g\ b \bmod P)\ a \bmod P$

Bob computes $g\ b\ a \bmod P = (g\ a \bmod P)\ b \bmod P$

As can be seen, both the values are same, so Alice and Bob have a same secret key, which can be used for ciphering or deciphering purposes.

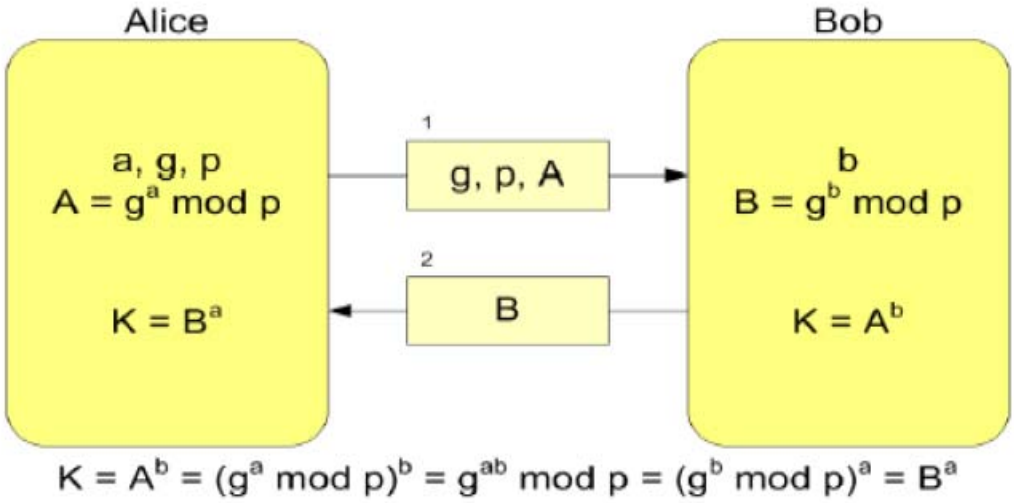


Figure 4-3: Diffie–Hellman key exchange algorithm [from Wikipedia.org]

4.1.2.3 Zeroization

This is one of the security functions needed for FIPS certification. It is a process that involves zeroization of all the CSPs in the case of any physical security breach. It can be manual or

automatic depending on the FIPS security level. After zeroization, on startup the CM goes through self integrity and known answer tests. The CM also goes through the key establishment process, since all the keys have been zeroized during the process.

4.1.2.4 Known Answer Tests

The known answer test (KAT) is used to test the integrity of the software implemented on a CM. In our case, we use AES and Diffie–Hellman algorithms, so at start up the values from the algorithm will be tested using a KAT. In the zeroization process, only the CSP values are zeroized, the software remains intact and so do the KATs. Thus, on startup, the processor will pick up the memory values from the AES and Diffie–Hellman algorithms and compare them with pre-calculated values. If this passes, the CM can go ahead and perform all other functions, otherwise it will go into an error state.

4.1.2.5 Start Up Test

This test is done to check if the software stored in the memory is intact. To do this, we use a check sum algorithm, in which the values stored in memory are taken as a block of data and are input into an algorithm and compared to an output value before shutting down the system. If these two values match, then that means the software has not been tampered with and the microcontroller can move ahead to do other functions.

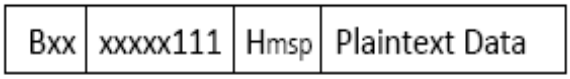
4.2 Data Flow

A unique data flow diagram was designed for this project. Figure 4-4 shows the flow of packets as the packet moves from one module to another. Table 4-1 below provides values for the identifier, which differentiates between different packet types.

Table 4 -1 Identifier Fields and their corresponding payloads [*Kanani*, 2009]

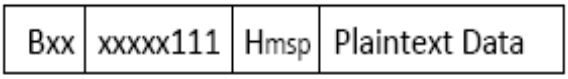
Identifier Field	Direction	Value of Data Field
XXXXX111	Master ⟷ Crypto	Plain Text Data
XXXXX100	Master ⟷ Crypto	Ciphered Data
XXXXX000	Master ⟶ FIPS	Empty packet for inserting key
XXXXX001	Master ⟷ Crypto	Packet #1 containing key material
XXXXX010	Master ⟷ Crypto	Packet #2 containing key material
XXXXX011	Master ⟷ Crypto	Packet #3 containing key material
XXXXX101/110	NA	Reserved

Step 1:



This is the first step in the key building process and it initiates the key exchange. Bxx is defined as the address of the receiver node; xxxx111 is the identifier bit that indicates that this is a plain text data; Hmsp is the header used for Master. In this step, the Master needs to cipher some data, so it sends the data to CM for ciphering.

Step 2:



The CM receives the packet and after looking into the values of the identifier bit (xxxxx111), it realizes that it contains plain text data that need ciphering. Bxx helps CM to know where the packet is coming from and thus helps in finding the appropriate key for that node. After checking its flag value, the CM realizes that it does not have key for ciphering this packet as there has been no key exchange process started yet. The CM sends the same packet back to the Master without any modification.

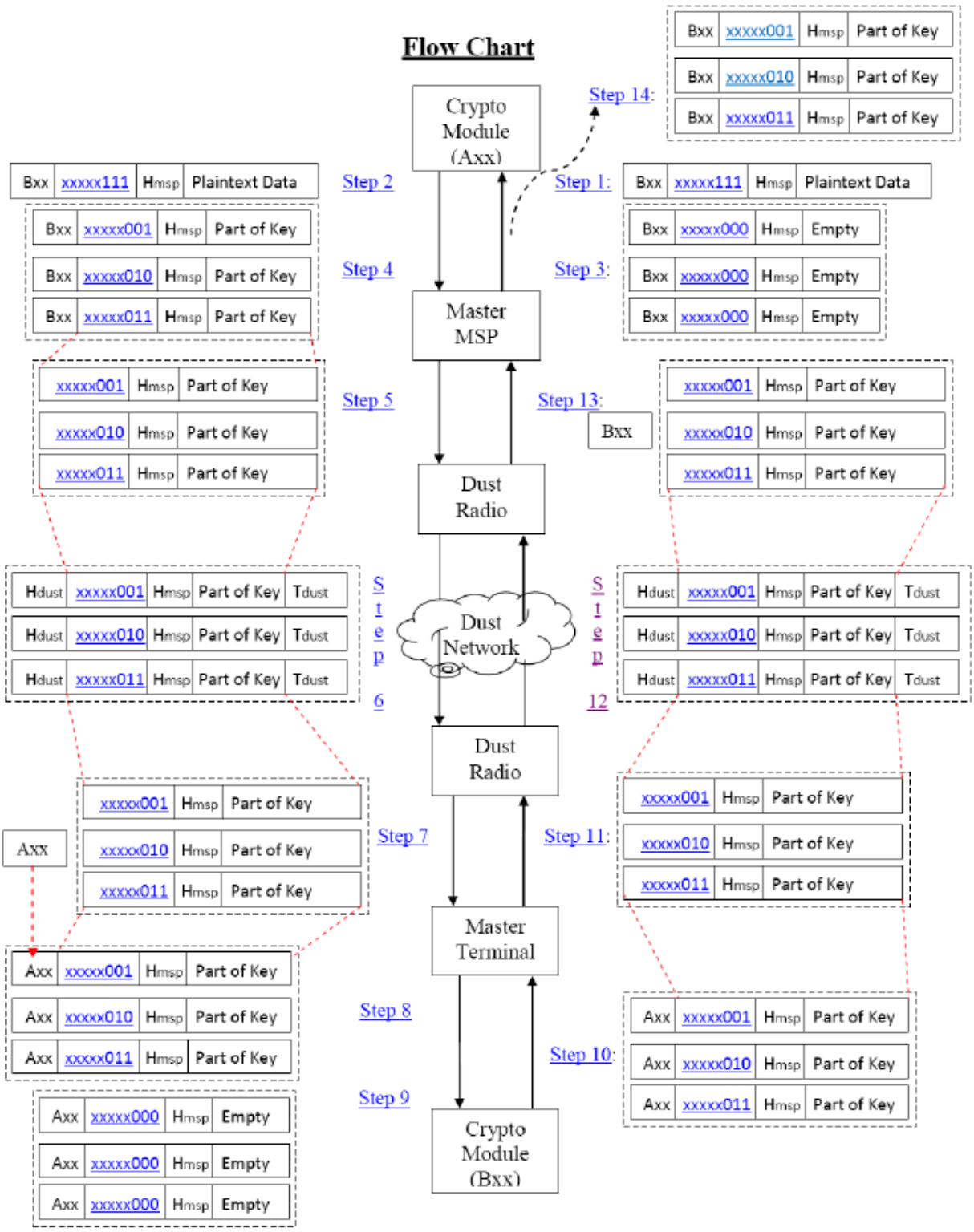
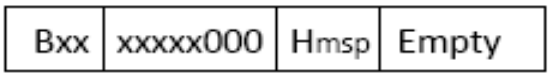


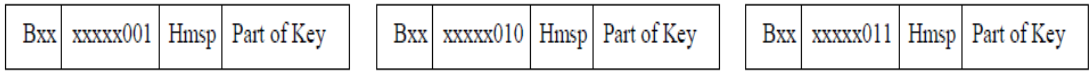
Figure 4 -4: Flow chart depicting the flow of packets [from *Kanani*, 2009]

Step 3:



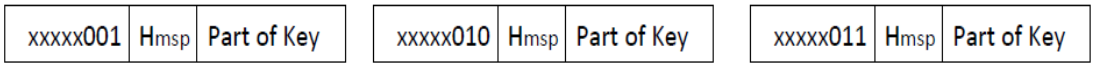
The Master is expecting a ciphered packet from the CM, but instead it receives the same packet it sent and thus realizes that CM does not have the key to cipher the packet. The Master starts the key generation process by sending three (this number can be modified) empty packets to the CM with identifier bits as xxxxx000 indicating that the packet is empty.

Step 4:



The Master sends the above packets to the CM, where Bxx again represents the address of the Master terminal. Packet 1 is identified with identifier xxxxx001 and, similarly, Packets #2 and 3 are identified with xxxxx010 and xxxxx011, respectively. As the CM receives the empty packets for key generation, it start generating the partial key using the Diffie–Hellman key exchange algorithm and sends the three packets back to Master after stuffing in the key it generated.

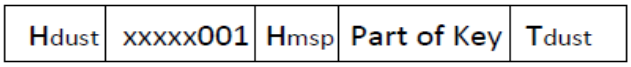
Step 5:



In this step, the Master receives the key material sent by the CM and sends this to Dust Radio for transmission out onto the Dust Network. Before the Master sends the packet out on the Dust Network, there is a small modification it makes, which helps in saving transmission bandwidth and avoiding the sending of redundant information: the Master strips off the address field Bxx from the packet. The reason for this is that, at the other end, the Dust Radio does not require knowledge of its own address, but it does require information on where the packet is coming from. Thus, it may sound logical to strip off the receiver address and instead attach the

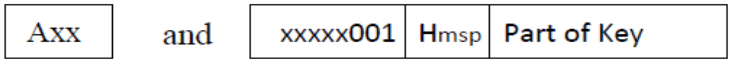
transmitter address (Axx). Now when the packet is received at the other end by the Dust Radio and is transferred to the Master and finally to the CM, it will know where the packet is coming from. However, there is another way by which this information can be achieved at the receiving end (explained in Step 7), so by not sending this information we can save bandwidth on the communication channel, which is really important.

Step 6:



Hdust stands for header attached by the Dust Radio, Tdust is the trailer attached by Dust radio, and xxxx001 implies this is Packet # 1. As the packet is transmitted out of the Master to the Dust Network, it requires a header and trailer to reach the specific destination. The header and trailer contain useful information that will help in deciding the route for the packet and the destination address. At the receiving end, these packets are collected and forwarded to Dust Radio for further processing.

Step 7:



In this step, Axx is the address of the transmitter node. Every node in the network has a unique address, which helps in identifying different nodes. The packet transmitted over the Dust Network contains a header and trailer that are stripped off before forwarding them to Master.

Along with the packet, Dust Radio also sends information about the transmitter node from where the packet originated. Since this information (Axx) was easily available from Dust Radio at the receiver end, this part of the packet was not included while transmitting, which saves some useful bandwidth in the Dust Network.

Step 8:

Axx	xxxxx001	Hmsp	Part of Key

As mentioned in Step 7, the Master receives two pieces of information from Dust Radio: the transmitter address Axx and the packet containing key information. The Master then combines these pieces and sends the combined packet to CM. Again, xxxxx001 represents the first packet and next two with identifiers xxxxx010 and xxxxx011 follow.

Step 9:

Axx	xxxxx000	Hmsp	Empty

As the Master sends the packets containing key information to CM, the CM understands that the key generation process has been initiated. The Master also sends three empty packets and, as in Step 3, the CM receives these three empty packets and starts generating 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

In this step, Axx stands for the address of the transmitter, which is now going to be the receiver. xxxxx001 indicates that this is Packet # 1 containing the key material, xxxxx010 represents Packet # 2 and, xxxxx011 represents Packet # 3 containing key material. As the CM receives the empty packets from the Master, it starts generating other half of the key using the Diffie–Hellman key exchange algorithm, which is then stuffed in to the packets and transmitted over Dust Radio.

Step 11:

xxxxx001	Hmsp	Part of Key

xxxxx010	Hmsp	Part of Key

xxxxx011	Hmsp	Part of Key

In this step, the Master terminal receives the packets from the CM containing the key information. As in Step 5, the Master terminal strips off the address field Axx; the reason for this is the same as explained in Steps 5 and 7. The packet after collection is then forwarded to the Dust Radio for transmission out onto the Dust Network.

Step 12:

Hdust	xxxxx001	Hmsp	Part of Key	Tdust

After receiving the packets from the Master terminal in Step 11, Dust Radio transmits these packets through the Dust Network. Hdust stands for the packet header, Tdust stands for the packet trailer, and Hmsp implies header for the Master terminal. These headers and trailers are important as they help in deciding the route of the packet through the Dust Network as explained in Step 6 above.

Step 13:

Bxx

and

xxxxx001	Hmsp	Part of Key

In this step, Bxx is the address of the receiver node, which can now be addressed as a transmitter. Dust Radio receives the packet from the Dust Network and strips off the header and trailer from the packet before forwarding it to the Master terminal. Along with this packet, the Dust Radio also sends information about the address of the transmitter, i.e., Bxx. Thus, the receiver node now knows from where the packet has come.

Step 14:

Bxx	xxxxx001	Hmsp	Part of Key

The Master terminal receives two items from the Dust radio, Bxx, which is the address of receiver, and packet information containing the key material. The Master combines these two

pieces of information and sends this to the CM, which completes the entire key exchange process. The important thing to notice here is that the Master terminal does not sends the empty packet again as in Step 3, because it knows it was the one who initiated the key exchange process.

4.3 Test Bench and Implementation

This part of the chapter concentrates on the components of the test bench on which the system was implemented and tested, including the hardware and software components. Using images and screenshots, we also provide some testing results and output from the workbench.

4.3.1 Test Bench Components

The development of the project was done on general purpose development kits, i.e., the Texas Instruments MSP-TS430PM64. IAR Embedded Workbench was used as the software development platform. In-circuit emulators were used, i.e., the Texas Instruments MSP-FET430UIF, to connect the software and hardware components.

4.3.1.1 Microcontroller

After careful study of available low power microcontrollers on the market, the Texas Instruments MSP430F1611 was chosen for this project (Figure 4-5). The MSP430 family of ultra low power microcontrollers consists of several devices featuring different set of peripherals targeted for various applications [*Texas Instruments*, 2006]. The MSP430F1611 device features four low power modes, a 16-bit RISC CPU and 16-bit registers for high code and power efficiency.

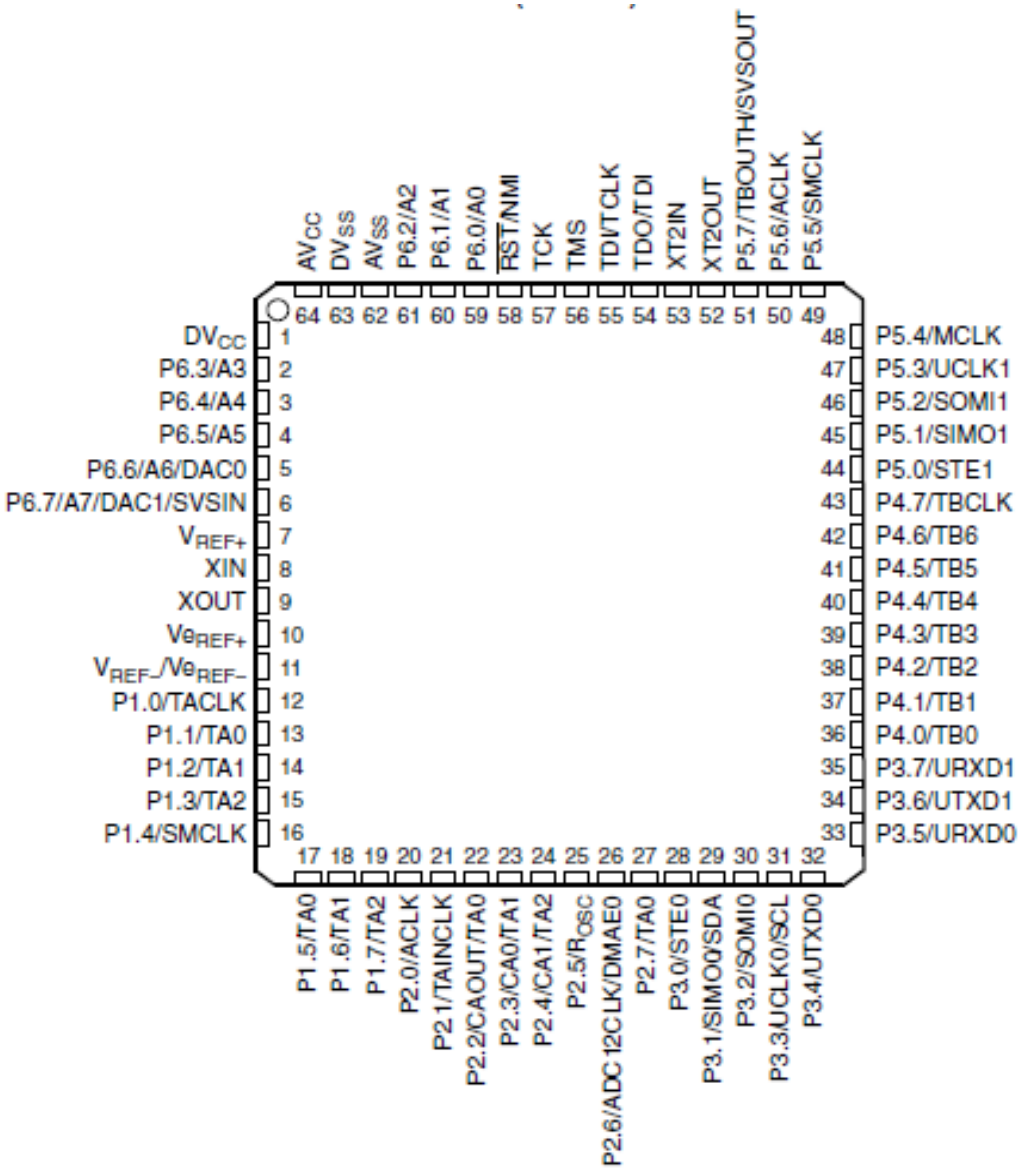


Figure 4 -5: Pin diagram for MSP430F1611 [from *Texas Instruments*, 2006]

Some of the important elements and features of MSP430 are described below:

- CPU: the MSP430 is a 16-bit RISC architecture CPU, which contains 16 registers that reduces instruction execution time. The register-to-register execution time is one cycle of the CPU clock [*Texas Instruments*, 2006]. Out of the 16 registers, four are special-purpose registers and the remainder are all general purpose. Special-purpose registers include Program Counter, Stack Pointer, Status Register, and Constant Generator.

- Bootstrap Loader (BSL): the MSP430 bootstrap loader allows the user to program the RAM or flash using a UART serial interface. The access to BSL to can be controlled using a user defined password.

- Direct Memory Access Controller (DMA): this is a very important component of this microcontroller as it helps in moving data from one memory location to another without CPU intervention. This feature improves the throughput of the peripheral devices and helps power conservation by allowing the CPU to remain in sleep state.

- Hardware Multiplier: there is a separate module for high level multiplication in the MSP430. This module can perform all 8-bit and 16-bit multiplication operations, including with signed and unsigned numbers.

4.3.1.2 Development Kits

We have used the 64-pin MSP-TS430PM64 as a target board for the code development. Made by Texas Instruments, it is a zero-insertion force (ZIF) socket target board used for programming and debugging the MSP430 through the JTAG interface or the SPY BI–Wire, which is a JTAG protocol developed by Texas Instruments. Figure 4-6 shows the PCB layout of this target board. The figure shows some important connections, such as Jumper J7, which is used for measuring the current consumption by the microcontroller, and J6, which is used to connect or disconnect the LED.

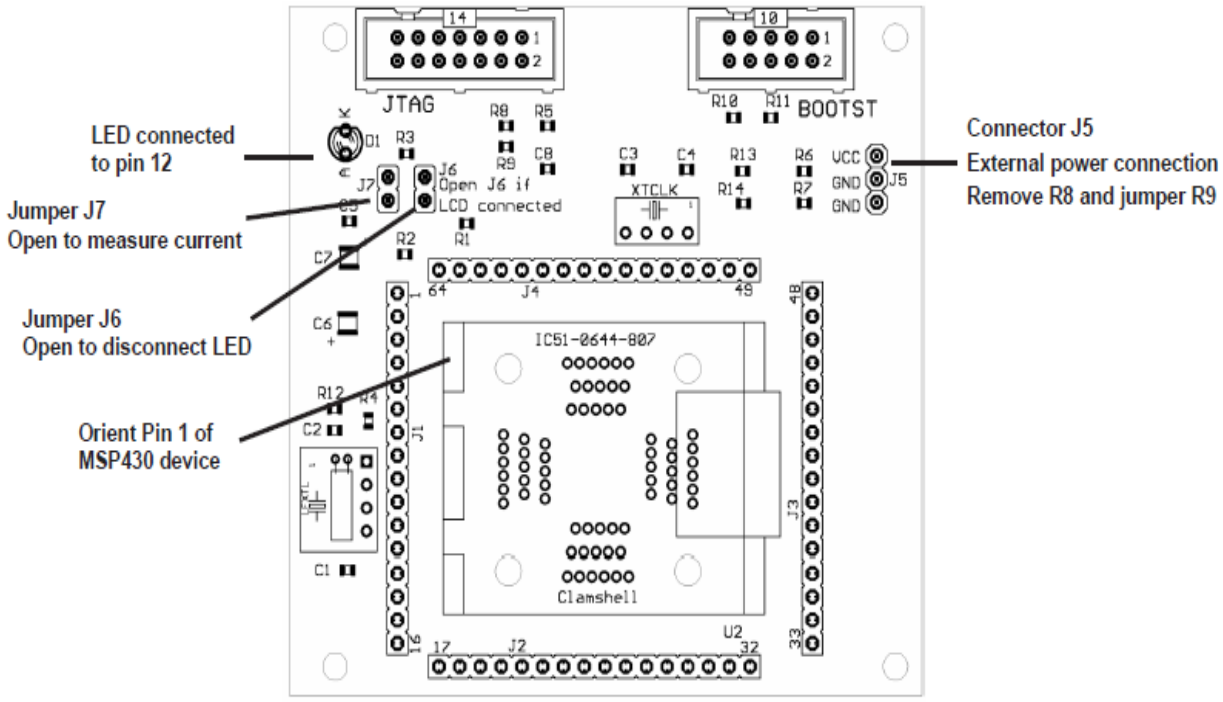


Figure 4 -6: MSP-TS430PM64 target socket module [from *Texas Instruments*, 2009]

4.3.3.3 Programming Interface

We have used the MSP-FET430UIF (Figure 4-7), which is a powerful flash emulation tool for application development on the MSP430 [*Texas Instruments*, 2009]. This tool includes a USB debug interface that can be used to program the MSP430 through the JTAG interface and can connect a flash-based MSP430 MCU to a PC for real-time programming and debugging.

4.3.3.4 IAR Embedded Workbench

We used the IAR Embedded Workbench to develop the firmware for the project. IAR software includes a C/C++ compiler, assembler, linker, text editor, and C-SPY Debugger in an integrated development environment (IDE).

Figure 4-7: MSP-FET430UIF MSP430 USB-Debug-Interface

IAR Workbench has the capability of building very efficient and reliable flash code for MSP430. It can output files in various formats such as .a43 and ihex, which can be used to download the firmware on MSP430 using any target board.

4.3.2 Implementation

As described in the above sections, the system implementation includes IAR Embedded Workbench, target boards with MSP430 microcontroller, and serial emulator. The two target boards are connected with each other via SPI bus as shown Figure 4-8. One of the boards acts as a Master MSP430 and the other as the CM used for encryption and decryption of data. The interface shown in Figure 4-8 is an exact copy of on the interface in the actual hardware (shown in Figure 3-4 above). Both of the target boards have an MSP430 microcontroller on them that can be programmed using the serial emulator. The serial emulator is connected to the PC via

USB, from where it gets powered up and also builds a connection between software and hardware platforms.

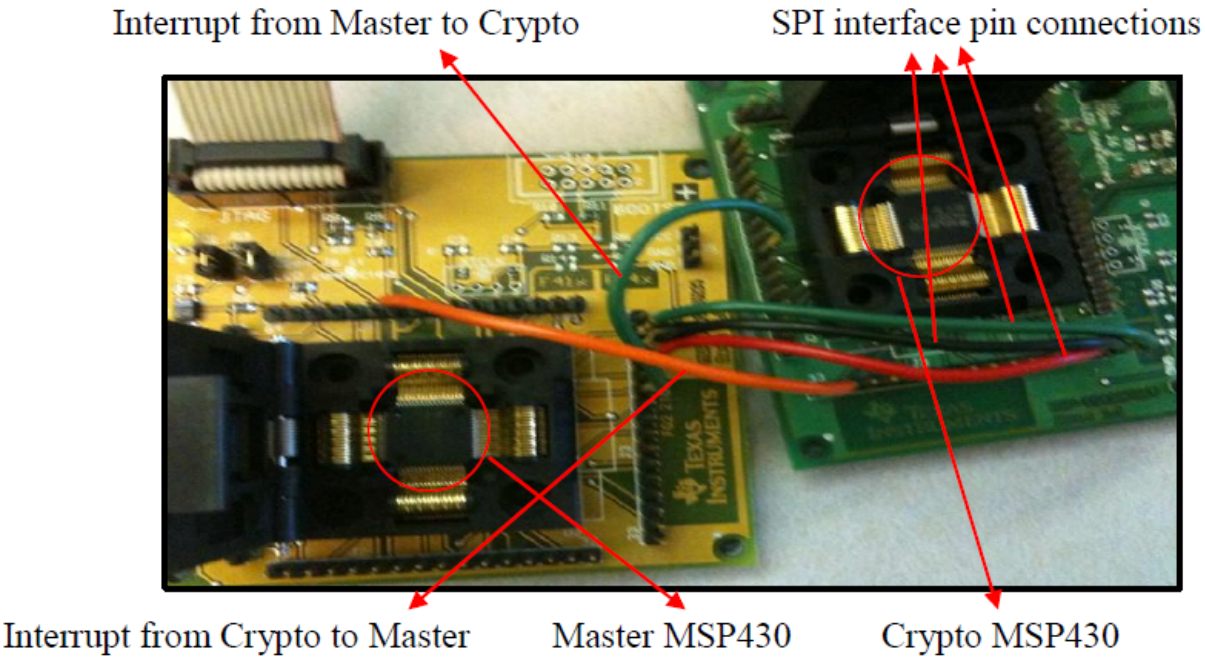


Figure 4-8: Experimental setup showing Master and CM interface.

IAR Workbench has an I/O window in which you can print various statements while the actual encryption and decryption is going on. This helps in understanding how the algorithm runs and the order of steps and also helps the programmer for debugging purposes. Figure 4-9 give a screen shot of the IAR software (I/O window) when the code implemented on the Master module was running.

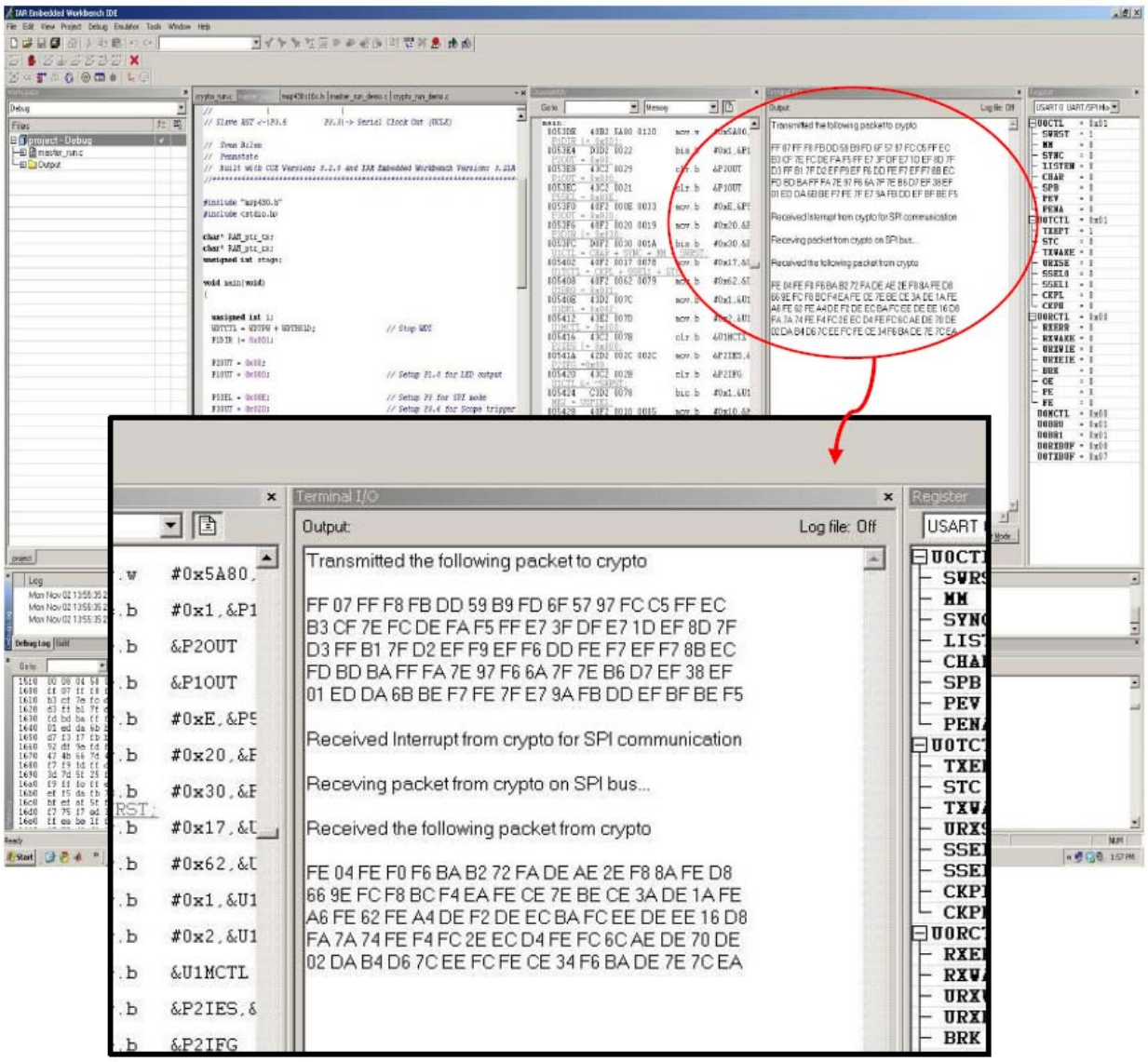


Figure 4-9 Screen shot of IAR Embedded Workbench with print statements

4.3.3 Test Inputs and Corresponding Outputs

As explained in the previous chapters, the CM uses identifier bits to recognize what kind of operation needs to be done on the packet received and also what kind of information packet is carrying. For example, if the packet sent has identifier bits as `07`, then that implies the Master is asking for the packet to be encrypted. The CM will read this information and perform the encryption algorithm. After encrypting the packet, the CM changes the identifier bit value to `04`

indicating that the packet is encrypted and then it is forwarded to Master. Another important thing to note here is that the first 16 bytes of the 80-byte packet sent by Master has header information and that part is not encrypted by the CM and is used for other purposes. The data below shows a sample run of CM using a specified key. Master sends the data with `07` identifier bit indicating plain text data, after receiving the packet CM module performs the encryption algorithm and send the data back to Master with `04` identifier bit indicating encrypted data.

Key used : `0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f`

Input from Master:

```
ff  07  ff  e9  fb  dd  59  b9  bd  6d  47  97  fc  c5  ff  ee
b3  cf  7e  fc  df  fa  f5  ff  e7  3f  ff  e7  1d  ef  8d  7f
d3  ff  b1  5f  d2  ff  f9  ef  f6  dd  fe  f7  ef  b7  8b  ec
fd  bd  ba  ff  fa  7e  97  f6  6e  7f  7e  fe  f3  ef  38  ef
83  ed  5a  6b  be  b7  ff  7d  e7  9a  fb  dd  ef  bf  be  f5
```

Output from Crypto:

```
ff  04  ff  e9  fb  dd  59  b9  bd  6d  47  97  fc  c5  ff  ee
70  9b  63  d4  30  24  3d  70  d0  c6  91  56  13  1f  d8  bf
0a  36  ec  cc  a0  28  a8  2b  21  64  de  99  a7  bc  1d  5d
c7  b7  f1  47  bc  84  ec  20  a4  bb  7b  59  c6  26  6a  2f
be  84  00  35  45  57  d2  1a  2f  b6  9d  de  f0  3c  63  f0
```

Now suppose that the same packet has to be decrypted at the other end. The Master sends the packet to CM with identifier bit set as `04` and the CM will again read this and decrypt the packet using the key. When sending the packet back, the CM will change the identifier bit to `07` indicating that it contains plain text data now.

Input from Master:

```
ff  04  ff  e9  fb  dd  59  b9  bd  6d  47  97  fc  c5  ff  ee
70  9b  63  d4  30  24  3d  70  d0  c6  91  56  13  1f  d8  bf
0a  36  ec  cc  a0  28  a8  2b  21  64  de  99  a7  bc  1d  5d
c7  b7  f1  47  bc  84  ec  20  a4  bb  7b  59  c6  26  6a  2f
be  84  00  35  45  57  d2  1a  2f  b6  9d  de  f0  3c  63  f0
```

Output from Crypto:

```
ff  07  ff  e9  fb  dd  59  b9  bd  6d  47  97  fc  c5 ff  ee
b3  cf  7e  fc  df  fa  f5  ff  e7  3f  ff  e7  1d  ef 8d  7f
d3  ff  b1  5f  d2  ff  f9  ef  f6  dd  fe  f7  ef  b7 8b  ec
fd  bd  ba  ff  fa  7e  97  f6  6e  7f  7e  fe  f3  ef 38  ef
83  ed  5a  6b  be  b7  ff  7d  e7  9a  fb  dd  ef  bf be  f5
```

Chapter 5
Power Analysis of the Module

In this chapter we discuss the power analysis of the CM developed in this project. This chapter is divided in to three parts. First, we discuss the need for low power consumption and why it is an important factor to consider in WSNs. Next, we describe methods to control the power consumption in the CM. Finally, we show some results from power analysis of the module and compare these results with other similar modules.

5.1 Need for Low Power Consumption

As explained in Section 1.4, with the increased use of embedded system in various industries like medical, defense, automation, consumer electronics, and many others, the need for low power consumption microcontrollers is increasing. In the current project, low power becomes even more important as the system is self powered and uses energy harvested from vibrations to generate power.

5.2 Power Optimization

There are different methods by which power optimization can be achieved, such as adjusting the voltage supplied, the frequency of operation of the microcontroller, optimization of the code, and using low power modes of the microcontroller. All of these factors are described below:

5.2.1 Voltage Supplied

The voltage supplied to the MSP430 is very important as it controls the current and the power consumption. As shown in Figure 5-1, MSP430 supply current varies linearly with input voltage, so operating the system at low voltages reduces the input current and, hence, overall power consumption.

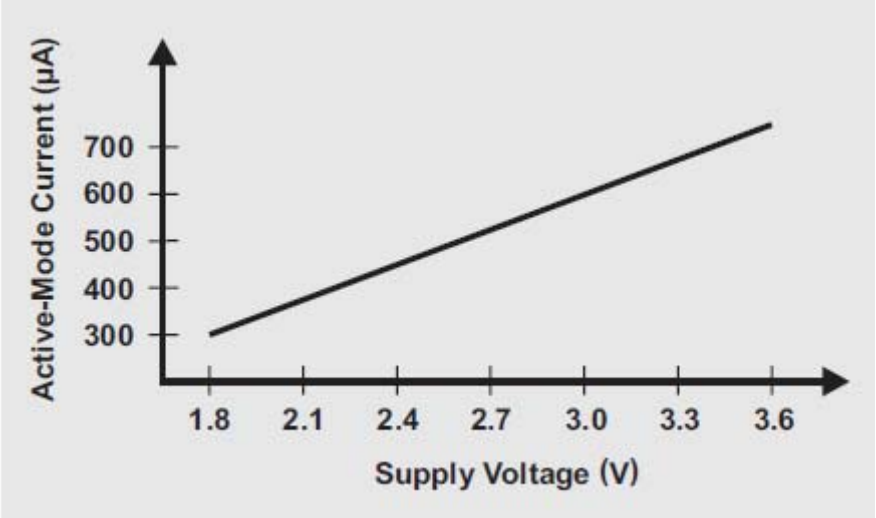


Figure 5-1: MSP430 supply current versus supply voltage [from *Day*, 2009]

In order to operate the system at low voltage, there is a linear regulator requirement which can control the voltage supplied to the system but adding a linear regulator between MSP430 and input-voltage source to increase the battery life might be contradictory because of two reasons:

- All power supplies have a quiescent current at no load that sinks current from battery to the ground.
- Power supplies have less than 100% efficiency.

To make the point more clear, let us consider two test cases. The first is to operate directly from the battery voltage and other is to insert a linear regulator between MSP430 and battery. According to *Day* [2009], it can been shown that, even after considering linear regulator efficiency and quiescent current loss, the case with linear regulator is much more efficient than the case without.

Figure 5-2 shows the two cases mentioned above. System 1 operates directly using two alkaline AA batteries, so all power supplied by the batteries is available to MSP430 as there is no loss

due to quiescent current or linear regulator efficiency. System 2 uses a TPS780XX linear regulator with an average efficiency of 90% and quiescent current loss of 500 nA.

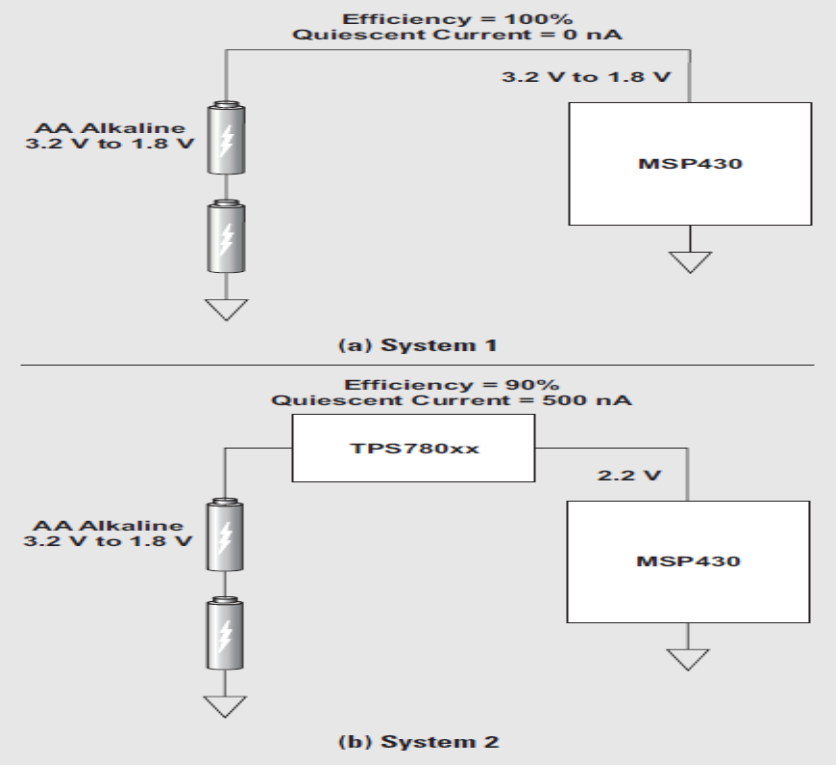


Figure 5-2: System configuration with and without linear regulator [from *Day*, 2009]

When the battery voltage is more than 2.2 V, System 1 consumes more current as compared to System 2 because, as shown before in Figure 5-1, the MSP430 operating current is a linear function of input voltage. System 2 in Figure 5-2 consumes a constant current as the linear regulator maintains a constant voltage of 2.2 V. As the battery voltage drops, both systems consume the same current at 2.2 V and below. There is additional current consumption of 500 nA by System 2 because of the quiescent current of the linear regulator (Figure 5-3).

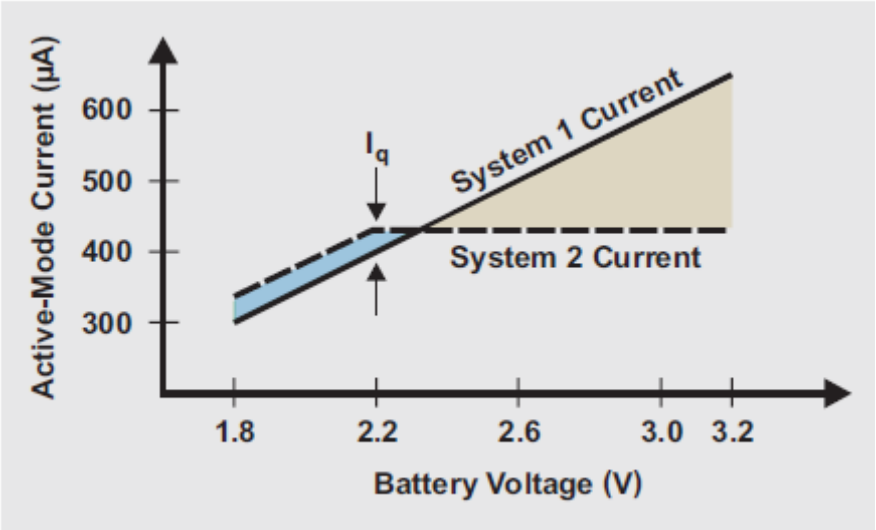


Figure 5-3: MSP430 current consumption versus supplied voltage [from *Day*, 2009]

The two cases mentioned above were implemented and it was found that System 1 operated for 223 hours, whereas System 2 with linear regulator operated for 298 hours. Hence, the addition of a linear regulator increased battery life by 30% [*Day*, 2009].

5.2.2 Frequency of Operation

Microcontroller frequency of operation affects the power consumption of the MSP430. The greater the microcontroller frequency, the less time it spends in active mode and lower will be the power consumed. However, for higher frequency we need to supply higher voltage to the microcontroller and, as discussed in Section 5.2.1, higher voltage means higher current and power consumption. So, depending on the system requirements, an equilibrium state can be found. If the system requires low execution time, then the microcontroller can be operated at maximum frequency; on the other hand, if the system requirement is low power then microcontroller frequency can be reduced.

Figure 5-4 shows the supply voltage required by MSP430 at different frequency values.

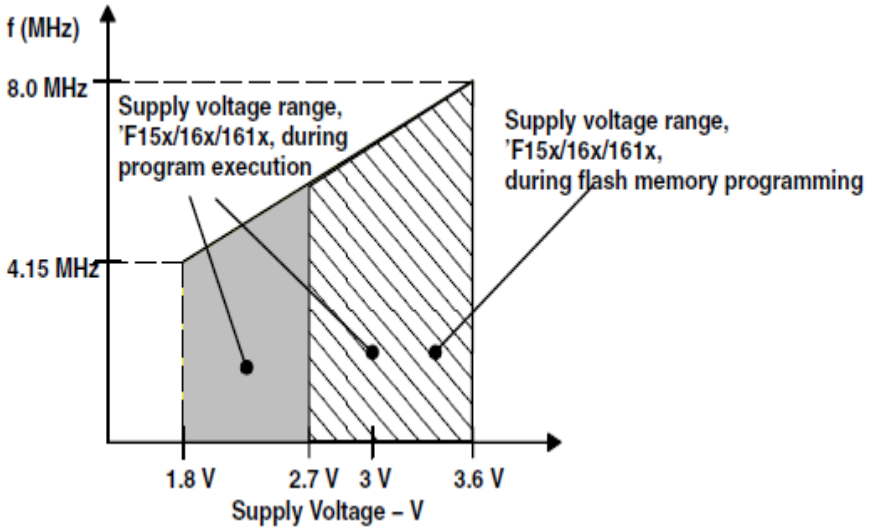


Figure 5-4: Frequency versus supply voltage for MSP430F16X [from Texas Instruments, 2009]

5.2.3 Low Power modes of Microcontroller

The MSP430 has several operating modes in which it consumes different amounts of power. The MSP430 has one active mode and five software-selectable low power modes of operation. An interrupt function can be used to wake up the MSP430 from any of the low power modes, service the request, and go back into the low-power mode. The list below provides all the operating modes that can be configured by software:

- Active mode: In this mode all the clocks and different peripherals are active.
- Low Power Mode 0 (LPM0): In this mode the CPU is disabled along with MCLK.
- Low Power Mode 1 (LPM1): CPU and MCLK are disabled along with DCO dc generator.
- Low Power Mode 2 (LPM 2): In this mode CPU, MCLK and SMLK are disabled but DCO is active.

- Low Power Mode 3 (LPM3): ACLK is active in this mode, whereas the CPU, MCLK, and SMLK remain inactive like LPM2.
- Low Power Mode 4 (LPM4): In this mode all the clocks are disabled along with the CPU, so this mode consumes the least energy.

In practice, there are three common modes used (Table 2-1): Active, LPM3, and LPM 4.

5.2.4 Code Optimization

Code optimization also helps in power conservation of the microcontroller. The compiler tool in the IAR Workbench includes an optimization function that improves the execution speed and reduces the size of C/C++ programs by performing functions like rearranging statements, simplifying loops, and allocating variables in registers.

5.3 Power Analysis Results

After keeping all the points mentioned above (Section 5.2) in mind, a system was designed to minimize the power consumed by the module. In this system we are using MSP430F1611 as the microcontroller. For maximum power optimization, the microcontroller use a supply voltage of 1.8 V in order to consumes minimum power. At 1.8 V, the microcontroller can run at a maximum frequency of 4 MHz. The CM was kept in LPM4 mode during the time no operations were required. Code optimization was implemented using the optimization tool of IAR Workbench.

5.3.1 Calculations

Below are the experimental values for power consumed by different cryptographic functions such as encryption, decryption, and key generation.

- **Encryption -** AES 128 bit encryption was carried out using the MSP430 at 1.8 V and 4 MHz. The total number of cycles was calculated using CYCLECOUNTER variable provided by IAR Workbench.

  Instantaneous current consumption in active mode: **517 µA**

  Instantaneous power consumption in active mode = $V \times I$ = 1.8 V × 517 µA = 930.6 µW

  Total number of cycles used = ~40958 cycles

  Total time consumed = ~40958 × 0.25µs = 10239.5 µs (frequency = 4 MHz)

  Total energy consumed for decrypting 64 bytes is = ~930.6 µW × 10239.5 µs = ~**9.528 µJ**

  Energy consumed per byte = **0.149 µJ/byte**

- **Decryption -** AES 128 bit decryption was carried out using the MSP430 at 1.8 V and 4 MHz.

  Instantaneous current consumption in active mode: **514 µA**

  Instantaneous power consumption in active mode = $V \times I$ = 1.8 V × 514 µA = 925.2 µW

  Total number of cycles used = ~49000 cycles

  Total time consumed = ~49000 × 0.25 µs = 12250 µs

  Total energy consumed for decrypting 64 bytes is = ~925.2 µW × 12250 µs = ~**11.33 µJ**

  Energy consumed per byte = **0.177 µJ/byte**

- **Key generation-** Diffie–Hellman was used for generating the key using MSP430 at 1.8 V and 4 MHz.

  Instantaneous Current consumption in active mode: **518 µA**

  Instantaneous power consumption in active mode = $V \times I$ = 1.8 V × 518 µA = 932.6 µW

  Total number of cycles used= ~36,800K cycles

Total time consumed = ~36,800K × 0.25 µs = ~ 9.2 s

Total energy consumed for decrypting 64 bytes is = ~932.6 µW × 9.2 s = ~**8.57 mJ**

5.3.2 Analysis of Results

We now calculate the overhead due to cryptographic functions on the total power consumed by the system. According to *Karri and Mishra* [2003], the amount of energy consumed for securely transferring 8 kB of data in a WSN system is 1164 mJ, which includes energy used in cryptographic computation and communication purposes. Cryptographic functions, which include key generation, data encryption, and authentication, consume 7.7% of the total energy. Energy consumed for communication purposes for transferring 8 kB of ciphered data was 1074 mJ (92.3 %).

Using the results from *Karri and Mishra* [2003], total energy consumed by our system for transferring 8 kB of data can be calculated as follows:

Energy consumption for encryption of 8 kB of data = 0.149 × 8000 µJ

Energy consumption for decryption of 8 kB of data = 0.177 × 8000 µJ

Energy consumption for key generation (assuming 2 key refresh) = 8.57 × 2 mJ

Total Energy consumed for cryptographic computation

= 0.149 × 8000 µJ + 0.177 × 8000 µJ +8.57 × 2 mJ = **19.74 mJ**

Using the above information, we can predict the energy consumed by our system for transfer of 8 kB of data will be 1074 mJ + 19.74 mJ = **1093 mJ**, of which only < **2%** of energy will be used for cryptographic computation and remainder is for data communication.

5.3.3 Comparison of Results

We now compare the results mentioned in above to other similar modules developed. The test bench used with different modules depends on the requirements of the system, which also affects the energy consumed. Table 5-1 shows the comparison between various modules with a description of their test bench and type of operation. The first entry is the module developed for this project as mentioned. The second entry refers to a WSN developed on Atmel Atmega128L microcontroller running at 4 MHz [*Wander et al.*, 2005]. The last entry refers to module implemented on StrongArm SA-1110 processor running at 206 MHz [*Karri and Mishra,* 2003]

The energy consumed by the key generation algorithm was also compared, but because of the vast number of key generation algorithms available, it was difficult to find similar modules for comparison. We have used the Diffie–Hellman key exchange algorithm for MSP430 running at 4 MHz and 1.8 V power supply. Another module developed by Sun Microsystems Laboratories used an MSP430 and Elliptical Curve Diffie–Hellman (ECDH) algorithm for key generation. ECDH uses Elliptical Curve Cryptography (ECC), which is a much newer and more efficient method of key generation.

Table 5-1: Energy comparison for encryption/decryption for several cryptographic modules

Type of Operation	Test Bench	Energy Consumption
Encryption/Decryption	MSP430, 1.8 V, 4 MHz	0.149/0.177 µJ/byte
Encryption/Decryption	Atmel Atmega128L, 4 MHz	1.62/2.49 µJ/byte
Encryption	StrongArm SA-1110, 206 MHz	0.536 µJ/byte

The same module was tested for RSA algorithm, which is not as energy efficient as Diffie–Hellman and the results are shown in Table 5-2.

Table 5-2: Energy comparison for key generation for several cryptographic modules

Type of Operation	Test Bench	Energy Consumption
Key Generation (Diffie–Hellman)	MSP430, 1.8 V , 4 MHz	~8.57 mJ
Key Generation (Elliptical Curve Diffie–Hellman)	MSP430, 3 V, 8 MHz	5.35 mJ
Key Generation (RSA)	MSP430, 3V, 8MHz	45.3 mJ

As can be seen from Tables 5-1 and 5-2, the amount of energy consumed by the module developed in this project is much lower than the other such modules available. The energy consumed for key generation is a little higher because of the difference in efficiency of the algorithm used. If either the same or equally efficient algorithms are used, then the current system will be most power efficient.

Chapter 6
Conclusion and Future work

This chapter provides some of the concluding remarks about the project and future work that can be considered. The goal of this project is to implement the FIPS approved sensor nodes for use on Navy ships. We have developed a cryptographic module with a CM–Master interface for use in Impact-RLW sensor nodes and deployed on ships.

6.1 Conclusion

In the work presented herein, we have implemented a security framework for data generated by sensor nodes and their subsequent transmission over the wireless channel. We have proposed an algorithm for encapsulating packets as they move from one module to another in the system. Since the system developed in this project between two MSP430s is not restricted to any wired or wireless communication protocol, it is quite flexible.

We successfully developed this low power CM capable of performing NIST approved algorithms such as AES, Diffie–Hellman key exchange, and KATs. Power optimization was another important goal of the project, since the sensor nodes are self powered and use vibration energy for harvesting power. Various methods of power optimization were implemented including code optimization, low power modes of microcontroller, and variation of microcontroller frequency and supply voltage.

6.2 Future Work

To make the module commercially viable, additional work must be done, including testing on RLW boards and EMC testing.

6.2.1 Testing on Impact-RLW boards

The code for the CM was developed and tested using Texas Instruments development kits and a JTAG emulator in a lab environment. To make the module ready for deploying on ships, the codes must to be transferred to the RLW sensor nodes and tested for all cryptographic functions. The power consumption of the microcontroller should again be tested and verified on RLW boards and the results should match the values mentioned in Chapter 5.

6.2.2 EMI/EMC Testing

As mentioned earlier in Chapter 2, one of the FIPS requirements for the CM is to undergo EMI/EMC testing. After the module has been tested and verified on RLW boards, it should undergo testing in a harsh environment like setting up the senor network in high electromagnetic interference environment. All the cryptographic functions and operation should be verified in this environment and made sure that the readings are correct.

Bibliography

(n.d). *Wireless sensor Network Technology* [online]. Available from http://www.coalesenses.com/index.php?page=technology

Atmel (2008). Atmel 8051 Datasheet. *Atmel Datasheet for 8051*. Atmel.

Day, M. (2009). *Using power solutions to extend battery life in MSP430 applications.* Texas Instruments.

Damutz, B. (2008). *Diffie-Hellman Key Exchange* [online]. Available from http://en.wikipedia.org/wiki/File:Diffie-Hellman-Schl%C3%BCsselaustausch.svg

Gupta, V., and Wurn, M. (June 12, 2008). *The Energy Cost of SSL in Deeply Embedded Systems.* Sun Microsystems Laboratories.

Hill, J., and Culler, D. (2002). *"Mica: a wireless platform for deeply embedded networks"*.Micro IEEE, vol. 22, no. 6.

Impact-RLW Systems, Inc., (July 2, 2009). *S^5NAP InfoSensor User Manual.*

Kanani, J. (2009). *Embedded Based Cryptographic Module for Low Power Wireless Sensor Nodes Complying with FIPS 140-2*. M.S. Thesis. Pennsylvania State University.

Karri, R., and Mishra, P. (2003). *Optimizing the Energy Consumed by Secure Session- Wireless Transport layer Security Case Study*. Journal ACM, *Mobile Networks and Applications,* vol 8, no 2

Kiratiwintakorn, P. (April 15, 2005). *Energy Efficient Security Framework for wireless local area networks*. Ph.D. Dissertation. University of Pittsburgh.

Lettieri, P., Schurgers, C., and Srivastava, M. (1999). "*Adaptive Link Layer strategies for energy efficient wireless networking.*" *Wireless Networks,* vol. 5, no. 5, pp. 339–355.

Microchip (2009). PIC 16F87/88 Data Sheet. *Microchip*.

National Institute of Standard and Technology (2002, 12 03). *Security Requirements for Cryptographic Module.* Retrieved 2009, from www.nist.gov: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

NIST (October 22, 2009). *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program.*

NIST: L. E. Bassham (August 2008). *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.*

Polastre, J., Szewczyk, R., and Culler, D. (2005). *Telos: enabling ultra-low power wireless research.* Fourth International Symposium on Information Processing in Sensor Networks, IEEE .

Prasithsangree, P., and Krishnamurthy, P. (2003). *Analysis of energy consumption of RC4 and AES algorithms in wireless LANs.* Global Telecommunication Conference (pp. 1445–1449). Globecom'03', IEEE.

Texas Instruments (May 2009). TI MSP430F1611 SLAS368F. *MSP430F161X Datasheet.*

Texas Instruments, (Feb .2009). *MSP-FET430 Flash Emulation Tool (FET) User Guide*.

Wander, A., Gura, N., Eberle, H., Gupta, V., and Shantz, S. (2005). *Energy Analysis of Public Key Cryptography for Wireless Sensor Networks.* University of California, Santa Cruz.

Yamanouchi, T. (2007). *AES Encryption and Decryption on the GPU* [Online] Available from http://http.developer.nvidia.com/GPUGems3/gpugems3_ch36.html