Brigham Young University
BYU ScholarsArchive

All Theses and Dissertations

2009-04-16

State of Secure Application Development for 802.15.4

Janell Armstrong
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

 Part of the Computer Sciences Commons

BYU ScholarsArchive Citation

Armstrong, Janell, "State of Secure Application Development for 802.15.4" (2009). *All Theses and Dissertations*. 1776.
https://scholarsarchive.byu.edu/etd/1776

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

www.manaraa.com

THE STATE OF SECURE APPLICATION DEVELOPMENT FOR 802.15.4

by

Janell Armstrong

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

School of Technology

Brigham Young University

August 2009

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Janell Armstrong

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Michael G. Bailey, Chair
Date	C. Richard Helps
Date	Joseph J. Ekstrom

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Janell Armstrong in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Michael G. Bailey
Chair, Graduate Committee

Accepted for the Department

Ron Terry
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

ABSTRACT

THE STATE OF SECURE APPLICATION DEVELOPMENT FOR 802.15.4

Janell Armstrong

School of Technology

Master of Science

A wireless sensor network consists of small, limited-resource embedded systems exchanging environment data and activating controls. These networks can be deployed in hostile environments to monitor wildlife habitats, implemented in factories to locate mobile equipment, and installed in home environments to optimize the use of utilities. Each of these scenarios requires network security to protect the network data. The IEEE 802.15.4 standard is designed for WSN communication, yet the standard states that it is not responsible for defining the initialization, distribution, updating, or management of network public keys.

Individuals seeking to research security topics will find that there are many 802.15.4-compliant development hardware kits available to purchase. However, these kits are not easily compared to each other without first-hand experience. Further, not all available kits are suitable for research in WSN security.

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

This thesis evaluates a broad spectrum of 802.15.4 development kits for security studies. Three promising kits are examined in detail: Crossbow MICAz, Freescale MC1321x, and the Sun SPOT. These kits are evaluated based on their hardware, software, development environment, additional libraries, additional tools, and cost. Recommendations are made to security researchers advising which kits to use depending on their design needs and priorities. Suggestions are made to each company on how to further improve their kits for security research.

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

ACKNOWLEDGMENTS

Thank you to Dr. Michael Bailey. He has been my guide, my taskmaster, and my cheerleader for the past several years. I would not have been able to complete this thesis without his assistance in removing research obstacles, his abundant corrections to my faulty grammar, and to his miraculously fast, detailed reviews of my work.

Thank you also to my thesis advisors. Thank you to Mr. Richard Helps for his insights in embedded system development, his assistance in finding funding for hardware kits, and for allowing me to work as his research assistant. Thank you to Dr. Joseph Ekstrom for teaching that the answer to an imprecise question is always, "it depends;" I still cherish the day I asked a question and received a direct answer.

Thank you also to the other present and former faculty and staff. Dr. Gordon Romney served as my first chair before his retirement. Dr. Barry Lunt provided an example of perfection organization and attention to detail. The current lab manager Mr. Mark Bailey and the former lab manager "Mac" who always kindly provided a listening ear when I needed to chat, and who almost always were able to provide the cables and tools I needed for my research. Thank you to Ms. Ruth Ann Lowe and Ms. Lynda Richmond for their attentive copy editing of my final thesis. Thanks also to Dr. Ron Gonzales, Mr. Steve Renshaw, Dr. David Anthony, and Ms. Jane Cunningham for their encouragement.

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

Thank you to the School of Technology for funding my research and providing lab facilities. Thank you to the School of Technology, the Engineering and Technology Student Council, and Boeing for granting me scholarships. Thank you also to the Graduate Student Committee who helped fund my conference travels. Thank you to FedEx and to General Electric for permitting me to take time off as necessary for my thesis.

Thank you to my supporting cast, particularly to Crystal Bateman for assistance in selecting hardware, collaboration in reading research papers, and for co-authoring conference papers. Thank you to all the other lab folk for their commiserations and advice: Rob Broadbent, Lloyd Brown, Nathan Blackham, Erin Bourgeous, Lane Broadbent, Travis Marks, and those whose faces I remember but whose names I cannot recall.

Thank you to my parents, Marian and Tim Armstrong, and to my brother, Eric Armstrong, for providing food and shelter for several years. Thank you to Becky and Koy Rehme and to Veronica Anderson for providing sanctuaries away from the lab.

In many ways I grew up in the Information Technology Program at Brigham Young University - from my first freshman-level IT classes to the defense of my master's thesis. It is impossible to properly acknowledge everyone who has helped me during this phase of my life without exceeding the number of pages already written for this thesis, so I must offer my general gratitude. Thank you to the many people who smiled and nodded while I raved about my thesis successes and while I ranted about my troubles despite their lack of understanding what "eight-oh-two-dot-fifteen-dot-four" let alone "C" really are.

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

TABLE OF CONTENTS

www.manaraa.com

www.manaraa.com

www.manaraa.com

www.manaraa.com

LIST OF TABLES

www.manaraa.com

www.manaraa.com

LIST OF FIGURES

www.manaraa.com

www.manaraa.com

1 Introduction

Wireless sensor networks (WSN) are a key component to turn the dream of "smart homes" into reality. A smart home, home automation, or environment automation describes a system where appliances communicate with one another, users remotely control and monitor their home, the environment automatically adjusts to a user's preferences, and so forth. Technologists and authors have long dreamed of achieving this technology - from E.M. Forster's science fiction short story describing an automated living environment (Forster, 2001)(originally published in 1909) to Mark Weiser's journal article defining ubiquitous computing (Weiser, 1991).

A WSN is a foundational technology in home automation. WSN's establish a channel through which multiple devices may communicate with each other. WSN's provide a means to monitor an individual's activities and habits. WSN's facilitate the ability to actuate both software and mechanical events throughout the system environment.

Typical users – even those accustomed to the pervasive technologies of cell phones, MP3 players, and computers – may not understand the benefits of home or environment automation; after all, manual control of their homes serves them well. Why would individuals wish for their appliances to collaborate, their security system to be

www.manaraa.com

remotely accessible, their homes to monitor the occupants' health, and their house lights to automatically turn on and off?

Automation through a WSN allows individuals and families unprecedented peace of mind, environment control, and home optimization. An automated home may provide independence for elderly relatives while alerting their caregivers when something is amiss (e.g. to summon assistance in response to a fall-related injury). An automated home may enable a person with handicaps to control their surroundings with ease. An automated home may help reduce utility bills by optimizing a family's use of electric lighting where young children and busy adults often neglect simple energy optimizations such as turning the lights off as they exit a room.

1.1 Wireless Sensor Network Technology

A wireless sensor network includes a set of sensor-equipped embedded systems, known as motes, communicating via a wireless channel. Each mote typically collects local environmental data, and then shares the data with its neighboring motes. Interested entities (e.g. users, system administrators, and other motes) may harvest these data to observe and control the current state and history of the monitored environment. Entities – including motes – may issue commands to or requests from the networked motes that in turn may control the user's home environment (e.g. the lawn watering system). Data and mote commands are considered privileged and sensitive because divulging the data may be reveal habits, routines, and the state of that which the network is monitoring.

The application program of the WSN may observe network traffic or receive mote data to analyze, monitor, and control the entire system. This application, usually running

www.manaraa.com

from a desktop computer, often acts as a director of the system: interpreting mote data and instructing motes on how to act. Specialized motes may aid the application in system tasks; these motes are known by many names including 'bridge,' 'gateway,' 'sink mote,' and 'coordinator mote.' Utilizing a specialized mote to control the network security creates a single point of failure, thus it is frowned upon in drop networks (thousands of motes deployed in a hostile environment)(Gamage et al, 2006). In home automation, however, this practice may be encouraged as a central point for a user to interact with, configure, and control the system while assuming the home itself provides sufficient physical security to protect the specialized mote.

Motes may pass messages from one to another over multiple hops within a peer-to-peer network. These messages may consist of acknowledgements, routing commands, the exchange of data, commands, and responses to commands. Each mote may read, analyze, ignore, forward, or respond to each received packet.

1.2 Network Standards

Several wireless technologies may be used to implement a wireless sensor network. Researchers created an irrigation system that used Bluetooth to deliver water to specific sites (Yunseop, Evans, Iversen, 2008). WiFi has been proposed as a backbone to multiple mesh networks (Leal et al, 2007). ZigBee (an implementation of 802.15.4) is commonly implemented within environment monitoring projects such as the oft-cited Great Duck Island application (Kumagai, 2004) used ZigBee devices to monitor the heat levels within bird burrows. Of these IEEE 802.15.4-compliant standards are often used because the standard is designed specifically for low-power, mesh network devices.

www.manaraa.com

The Institute for Electrical and Electronics Engineers (IEEE) proposed and first ratified the 802.15.4 standard in 2003 for communication within wireless sensor networks (IEEE Computer Society, 2006). The 802.15.4 standard, part of the WPAN (Wireless Personal Area Network) working group, is specifically tailored to low-bandwidth, limited-battery devices communicating in peer-to-peer, ad-hoc connections. The ZigBee alliance, a consortium of over 250 member companies (ZigBee Alliance, 2008), developed and ratified the ZigBee specification that defines additional security mechanisms, routing protocols, and efficiency measure to 802.15.4 (ZigBee Alliance, 2004, ZigBee Alliance, 2006). Each of these standards as well as other common wireless network standards will be discussed further in Section 2.1.

1.3 Network Security Threats

This new technology introduces many benefits to its users, yet wireless sensor networks may also introduce new problems for its users. Like all wireless communication, a WSN is vulnerable to both passive and aggressive security attacks. Data transmitted through a WSN channel without security precautions may be intercepted or altered by unauthorized users. Security flaws may permit identify theft, cause system down time, and allow unauthorized users to control the network and its devices. Examples of potential security vulnerabilities include eavesdropping, service disruption, protocol attacks, etc. (See Chapter 2 for a further discussion on security threats and the research towards resolving the system weaknesses.)

The network owner must protect the network data by making it difficult to understand and by blocking unauthorized entities from entering the network.

www.manaraa.com

Furthermore, a balance between security and resource consumption must be achieved; a high-consumption security system may disable a WSN's ability to allocate resources to its core application, yet a low-cost security system may not sufficiently defend the network. Security tools must be developed to enable network owner to create a secure WSN.

1.4 Development Challenges

Researchers may study the effectiveness in preventing network attacks by studying WSN weaknesses and developing security fortifications. Typically, this research is discussed as theory defended by a comparison to similar methods within other wireless network standards or by data produced by software simulations. WSN software is not often studied on actual hardware due to scaling constraints, budgeting, or an assumption that simulation is sufficient to prove a security method effective.

Implementing a wireless sensor network on real hardware is not without its challenges. The exchange of large data packets between two motes is often restricted by narrow bandwidth. Debugging is difficult because of the nature of a distributed system. The mote software may be particularly constrained by the hardware's limited processor and limited RAM. An example of the limited mote resources is the Freescale MC13213 mote. This mote has only a 40 MHz, 8-bit processor; 60 KB flash; and 4 KB RAM (Freescale Semiconductor Inc., 2007z). These hardware limitations limit the speed, program size, application complexity, and security of the WSN.

Further challenges exist when the learning curve of existing hardware is taken into consideration. The available documentation and tutorials; accessible, active, online

www.manaraa.com

communities; open-source code and demonstration code each have an impact on the success of a developer in writing code. A developer is aided in debugging software by the hardware interfaces used to communicate with the device and by the device's buttons and LEDs. The cost of the hardware impacts the number of devices researchers can obtain and consequently influences the network scale. The existing Operating Systems and code may further assist a developer by adhering to an OSI-model of code by separating out the code necessary for activating an LED from the network routing code; thus enabling the new users to focus on development without perfecting their knowledge of the lower levels of coding.

Unfortunately, there is no "perfect solution" development kit available to student researchers. Most existing development kits lack one of the crucial components and may hinder success in security research.

1.5 Proposed Research

This thesis shall assess several WSN development kits for suitability to security development and research. The hardware of each kit will be described including its memory, processor, user interface and sensors, daughterboards, power supply, and housing. The development environment and software requirements will be described in enough detail to support the analysis. The available documentation for each kit will be described. The various kit configurations available for purchase and the expense of each will be noted.

The suitability of each kit will be measured against a security development scenario. The test scenario shall be a hypothetical implementation of a simplified version

www.manaraa.com

of SSL to create a secure channel between two motes and of a periodic key distribution. The number of necessary SSL packets to establish communication shall be reduced by eliminating certificates and by limiting each mote to a single encryption method. In addition, due to the large memory overhead of Public Key Infrastructure (PK) systems, these will also be excluded from consideration in this research. The justification for selecting this particular test scenario will be discussed further in Section 3.2.

The strengths and weaknesses of each development kit will be discussed in the context of the tools required to develop the test scenario. Conclusions will be drawn identifying what types of research and tasks for which each development kit is best suited, since one kit may be better suited to research in application security whereas another kit may be better suited to research in the implementation of encryption algorithms. Recommendations will be made on how to improve each kit to better suit it for security development in academic research.

www.manaraa.com

www.manaraa.com

2 Review of Literature

This chapter provides an overview of the state of the art of development and security within wireless sensor networks. The suitability of several wireless standards will be reviewed for WSN. The current state of 802.15.4 and ZigBee security is presented. WSN security threats are enumerated, and existing research is discussed.

2.1 Wireless Communication Standards

Wi-Fi, UWB (Ultra Wideband), Bluetooth and ULP (Ultra Low Power) Bluetooth, IEEE 802.15.4, and ZigBee have emerged as wireless protocols. Each protocol has its own strengths and weaknesses. This section considers each protocol for use within a wireless sensor network.

WSN communication standards should accommodate the expected hardware limitations of WSN motes and support peer-to-peer networking. Hardware constraints further require code with lightweight memory footprints and efficient algorithms. The network standard must allow for several anticipated scenarios: new motes may enter the network without warning; current motes may exit the network without warning (e.g. a battery dies or a fatal accident occurs); mobile motes may move throughout the network unpredictably; and motes may exchange data with any other mote within the network.

www.manaraa.com

Potential standards to WSN communication include Wi-Fi, UWB, Bluetooth, 802.15.4 and ZigBee.

2.1.1 Wi-Fi

The IEEE 802.11 standard (Wi-Fi) specifies network access to high-powered devices with high data rates. This standard is commonly used for WLAN connectivity; particularly when channeling Internet traffic. Typically, the 802.11 network topology consists of a base station communicating directly with child nodes and a single Internet or LAN connection. While less commonly implemented, Wi-Fi-enabled devices may operate in a peer-to-peer, ad hoc mode. The proposed 802.11s, will allow base stations to route internet traffic through a mesh network (Cherry, 2006); vendors have begun to release draft-compliant products in anticipation of the standard's completion in late 2009 (PacketHop Inc., 2008).

The widespread adoption of Wi-Fi provides a ready-infrastructure to home automation; however, the high price of the Wi-Fi chipsets and their very high power consumption lessens the appeal of Wi-Fi for WSN devices. Anis Koubaa and Mario Alves make a good argument and later demonstrated that Wi-Fi ought to be used as a backbone in a large-scale WSN (Koubaa, Alves, 2005; Leal et al, 2007). However, their 2007 demonstration shows that the 802.11 standard is inappropriate for local, inter-mote communication.

2.1.2 Ultrawideband

Ultrawideband (UWB, IEEE 802.15.3) is expected by its designers to become a USB cable replacement technology by transmitting data over many frequencies (similarly

www.manaraa.com

to the way gigabit Ethernet transmits over several wires). Standard ratification has been delayed because the design committee was divided into two factions. The UWB Forum argued for direct-sequence ultrabandwidth; whereas the WiMedia Alliance supported multiband orthogonal frequency-division multiplexing (OFDM)(Geer, 2006). Arguments for each communication method were based on their effect on radio interference with existing devices, ungoverned use of the frequency spectrum in Europe and Asia, power consumption, chipset expense, and backwards compatibility with USB 2.0 (Schilit, Sengupta, 2004; Jones, 2004; Goth, 2007).

Eventually, the UWB Forum yielded to the WiMedia Alliance (Leavitt, 2007). During the schism, the WiMedia Alliance – led by Intel – continued to seek alternate standardization for its technology from the ECMA International and International Standards Organization (ISO). WiMedia had also gathered endorsements from the USB Implementers Forum (USB-IF) and the Bluetooth Special Interest Group.

During the time without a unified UWB standard, companies began to create their own wireless USB solutions. Freescale, the original leader of the UWB forum, withdrew and developed its CableFree standard and attempted to take the technology straight to market through the 2006 Consumer Electronics Show (Geer, 2006; Jones 2006; Goth, 2007). Pulse~Link, at the 2007 Consumer Electronics show, demonstrated its own proprietary UWB technology for communication between multiple home entertainment devices and a data network (Pulse~Link, 2007). As of 2007, 15 other companies began to release their own wireless USB products including Belkin International, D-Link, Hewlett-Packard, and Intel (Leavitt, 2007).

www.manaraa.com

The divisions, uncertainty, and foreboding of future compatibility problems were sufficient to disqualify UWB as a communication standard within WSN's.

2.1.3 Bluetooth and ULP Bluetooth

Bluetooth (IEEE 802.15.1) was originally designed to replace cables between computer peripherals for the high-volume transfer of files and data. The Bluetooth standard allows for point-to-multipoint, ad hoc communication for inter-peripheral or desktop-peripheral communications. A Bluetooth piconet allows a single device to be the master of up to seven child devices. The master device communicates with each child in a round-robin fashion, and a child device may not communicate directly with another child device. In a Bluetooth scatternet a child may participate in multiple piconets, but the standard does not define any communication between piconets.

Ultra Low Power Bluetooth (ULP Bluetooth), on the other hand, is designed to enable communication between small devices. Nokia initially announced this Bluetooth-complimentary standard as WiBree in 2006. They proposed an adapted Bluetooth standard that featured 1Mbps bandwidth, peer-to-peer mesh networking, and Bluetooth-compatibility (Nokia Corporation, 2007a). In 2007, the Bluetooth SIG announced that it would adopt WiBree as part of the Bluetooth communication standard (Nokia Corporation, 2007b). Recently, CSR (a British silicon chip company) provided the first public demonstration of the technology (now called ULP Bluetooth). The demonstrated technology achieved packet transfers at 50 times the rate of standard Bluetooth and consumed 1/10 the power required by standard Bluetooth (CSR, 2008).

Bluetooth is an attractive choice with its inclusion of device type profiles, authentication and pairing protocols, and widespread adoption. End-users may find

www.manaraa.com

convenience in a home network that seamlessly connects with their existing Bluetooth WPAN. The "ultra low" power consumption of ULP Bluetooth even promises to enable small, battery-powered devices to standby for years (CSR, 2008).

Unfortunately, ULP Bluetooth build environments are unavailable to the public at this time. Further, Bluetooth does not inherently accommodate peer-to-peer networking. Therefore the standard is currently unsuitable for WSN applications.

2.1.4 802.15.4

IEEE 802.15.4 defines a mesh-networking, communication standard for very low data-rate, battery-limited devices (IEEE Computer Society, 2006). The standard defines the Media Access Control (MAC) layer and physical (PHY) layer of the OSI model. The key features of this standard are its minimal power consumption for extended battery life, simultaneous support of hundreds (and in some configurations, thousands) of devices, and peer-to-peer communication.

Several companies have sought to create similar technologies; three of these proprietary variations are particularly notable. SmartLab's Insteon implements the 802.15.4 standard while eliminating the need for a network controller and message routing (SmartLabs Inc, 2007). Likewise, Microchip's MiWi creates a light-weight version of 802.15.4 by eliminating all security measures and by limiting the network communication to 4 hops (Microchip Technology Inc, 2006). Zensys's Z-Wave is similar to 802.15.4 in functionality but is not 802.15.4-compliant (Z-Wave Alliance, 2007). At the time of this thesis neither academia nor industry had seriously adopted any of these variations.

www.manaraa.com

2.1.5 ZigBee

The ZigBee specification (ZigBee Alliance, 2007b), adds application framework and network layers, and some security measures to IEEE 802.15.4. This specification defines three types of motes and three network topologies. The specification also decreases power consumption by allowing developers to synchronize the duty cycle of the network motes.

Every ZigBee node must be one of three types of motes: Coordinator, Router, or End Device. The ZigBee Coordinator is responsible for network configuration and synchronization. Every network, regardless of topology, must have a Coordinator. Every network, regardless of topology, may only have one Coordinator. ZigBee Routers are responsible for the routing of network messages to their associated End Devices and neighboring Routers. The Coordinator also shares routing responsibilities with the Router nodes. ZigBee End Devices only communicate with their parent Routers.

ZigBee supports three network topologies: star, tree, and mesh. Regardless of network topology, every network type must have one and only one Coordinator and End Devices may only communicate with their parent node. The ZigBee star network is like a Bluetooth piconet, for neither make allowance for communication between the children of the network hub device (i.e. Coordinator). In a mesh and tree network, if a device's parent ceases to exist within the network the device will identify and join another Router within its range; if no new parent is available the device will be unable to rejoin the network. In a mesh topology messages between two out-of-range nodes are routed through the network Routers.

www.manaraa.com

ZigBee minimizes power consumption and prolongs the battery life by operating on a very low duty cycle of less than 1%, by transitioning devices from sleep state to an active state in 15ms, and by requiring only 30ms for a device to join the network (ZigBee Alliance, 2007a).

The device duty cycle of star and tree networks are defined by beacon mode synchronization. Coordinator transmits beacon frames to all its associated nodes which in turn route the frames to their associated nodes, etc. Following each of these frames is the active period in which devices may exchange messages by CSMA (Carrier Sense Multiple Access). This methodology requires a node to listen for network traffic and wait for the channel to be free. All nodes sleep during an inactive period and awaken for the next anticipated beacon frame.

The beacon frame is propagated throughout the network, so a mesh network may not operate in beacon mode. In non-beacon mode data transmissions may take place at any time and are not limited to the active period. All transfers take place during a free period determined by CSMA. Nodes participating in a mesh topology must have an increased duty cycle (and thereby increased battery consumption) because there is no defined active period.

These ZigBee specification constraints prevent the network from being robust because the entire network may fail if the Coordinator is cut off from the network. Relying on specialized devices to perform network tasks may interfere with a network's robustness. In a beaconing network the loss of the Coordinator would be detrimental because only the Coordinator sends the synchronization beacons. Similarly, in a star network the loss of the Coordinator would be detrimental because only the Coordinator

www.manaraa.com

may route messages. In a tree or mesh network, the loss of a Router may cut many devices off from the remaining network if no other Routers are within range of the lost Router's associated devices.

Unfortunately, despite the nearly ideal design of the ZigBee specification, the ZigBee Alliance has unintentionally hindered the adoption of the technology. One author succinctly stated, "ZigBee has feature-creeped far beyond its humble 'home RF' beginnings. That's not necessarily bad, reflecting an understandable migration of application aspirations to bigger and better things. . . On the other hand, with wireless it's always the more the merrier, and ZigBee feature creep leaves the door open for new low-end alternatives" (Cantrell, 2008). Additionally, ZigBee-compliant, open-source, security-enabled, stack source code is difficult to obtain. Closed stack code greatly increases the complexity of manipulating the security state machine to add additional measures or add evaluation and measurement algorithms.

2.1.6 Conclusions

IEEE 802.15.4 and its derivative, ZigBee, are equally appropriate protocols for the development of a WSN. Unlike Bluetooth and UWB, 802.15.4 and ZigBee allow for mesh networking. Unlike WiFi, 802.15.4 and ZigBee may be powered from a single set of batteries for weeks or months. Both 802.15.4 and ZigBee are specifically designed for devices with small microprocessors and limited memory.

Alternatively, researchers may find that either 802.15.4 or ZigBee is better suited to specific research topics. The 802.15.4 standard may be more accommodating to the advancement of WSN security research, as it defines only the basic conventions for security thus allowing the researcher to define a complete security system. ZigBee is

www.manaraa.com

appropriate for the design of secure WSN applications because it standardizes the network-layer among third party devices.

The remainder of this thesis will address both 802.15.4 and ZigBee as either may be selected for WSN research.

2.2 802.15.4 and ZigBee Security

Both 802.15.4 and ZigBee provide basic mechanisms necessary to implementing a security policy (IEEE Computer Society, 2006; ZigBee Alliance, 2007b). The 802.15.4 standard provides frames security, and ZigBee adds key transport, device management, and a permissions matrix. Unfortunately, each standard acknowledges that much of security necessary to WSN implementation is outside the scope of its standardization.

IEEE 802.15.4 is a standard focused on specifying the physical and data layers of WSN communication. It acknowledges that, "[Most] security architectural elements can be implemented at higher layers and may, therefore, be considered to be outside the scope of this standard" (IEEE Computer Society, 2006). The standard provides frame protection using a key shared between two devices or among a group of devices, and a security level may be associated within a request command frame. The standard also provides cryptographic mechanisms based on symmetric-key cryptography using a key provided by high layer processes. However, despite providing the basic mechanisms for a security policy, the standard declares that "the establishment and maintenance of keys are outside the scope of this standard."

The ZigBee specification further defines the Network and Application layer, and the more recent revisions have emphasized network security. The specification designates

www.manaraa.com

a specialized mote with controlling the network security, and it defines a master key from which all other keys are derived. The specification also provides a mechanism for key transport – that may only be encrypted if the key has already been defined on both the server and client motes. Within the specification, all master keys are either preinstalled or the key updating scheduling is left to the application layer. ZigBee-2007 also provides a configuration table for permissions to matrix what commands a mote may or may not heed.

The ZigBee-2007 specification includes many security mechanisms for key establishment, key transport, frame protection, and device management. The specification notes, "These services form the building blocks for implementing security policies within a ZigBee device." (ZigBee Alliance, 2007b) Particularly useful to a researcher developing a security policy are the mechanism provided to request a key, transport a key, switch a key, or inform a router than one of its children must be updated or removed from the network.

The security mechanisms of ZigBee-2007 do not negate the need for careful attention when developing a security policy. The specification rests on the assumptions in the safekeeping of keys, in the secure initialization and installation of keying material, and in the processing of the keying material. The specification also relies on the application to handle error conditions, detect and handle loss of key synchronization, and manage the expiration and updating of key. Indeed, the specification emphasizes, "Trust in the security architecture ultimately reduces to trust in the secure initialization and installation of keying material and to trust the secure processing and storage of keying material." (ZigBee Alliance, 2007b).

www.manaraa.com

2.3 Security Threats

Mote communication is inherently public because motes communicate through a wireless channel. Authorized entities have access to the network data while eavesdroppers have equal access to unprotected data; an eavesdropping device, Eve, may access the network traffic of a user, Alice, by merely sitting within the network broadcast range. Conversely, wired network is secured by tethering the system to a firewall or secured gateway, but a wireless connection has no such luxury. In a wireless network there is no authorized man-in-the-middle monitoring for suspicious traffic and blocking ill-doers. Further, passive access to wireless connections is virtually undetectable. A WSN is therefore susceptible to attack from unauthorized users. (Attack being here defined as reading, modifying, blocking, or adding to network traffic without permission from the network owner.)

Threats to information security are often classified within several subgroups relevant to security: integrity, confidentiality, availability, and authentication. This section touches on several of the issues pertaining to WSN threats. For a full threat analysis see the threat analysis report edited by Casaca and Westhoff (Girao et al, 2006).

Integrity verifies that data cannot be modified without authorization. Without any security whatsoever it is possible for a neighboring network to simply mistake a device for one of its own. However, an attacker may also utilize attack a system by exploiting a protocol weakness; thus establishing a man-in-the-middle mote that may read and alter messages between a client and a server without either device knowing about the intruding mote's presence. The integrity of a network may also be compromised when a mote accepts forged messages or replayed messages from an unauthorized device.

www.manaraa.com

Confidentiality prevents information from being revealed to unauthorized entities. Information is revealed to unauthorized entities when commands, messages, or keys are transmitted by plaintext. Messages without sufficient obfuscation – even when encrypted – may be easily understood by an observer when the packet is as simple as a "yes," "no," "on," or "off."

Availability ensures the network services and devices are able to communicate when needed. Service disruption attacks occur when an attacker disables the network communication by either preventing a signal from being received by the end device. An attacker may physically attack the device by jamming the signal or by physically tampering with the device; neither of these scenarios can be addressed by software. An attacker may also disrupt the network service by forming a DoS attack against a network device – by inundating it with so many messages that the device cannot respond to normal network traffic. A corollary of the DoS attack is called "sleep deprivation" (Stajano, 2002) in which an attacker forces a device to remain active and thereby draining its battery.

Authentication validates the communicating entities and prevents unauthorized entities from masquerading as a valid device or user. For example, Eve introduces a network-compatible, yet unauthorized, mote is to her neighbor's network. This unauthorized mote may now have the ability to forward the network traffic to Eve's desktop computer and it may have the ability to issue false data to the authorized network motes.

www.manaraa.com

2.4 Existing Security Implementations

The communication security of wireless sensor networks presents many unique challenges. Traditional methods of encryption with session keys and public keys are too resource heavy for the limited motes. In this section the desirable attributes for the security of a wireless sensor network are identified, and the state-of-the-art of 802.15.4 and ZigBee are evaluated. Emphasis is placed on security solutions that provide key initialization, key updating, and group management policies.

As previously discussed, motes have limited processor, memory, and power resources. While future motes may have more processor and memory resources, motes will likely always be constrained by their batteries (Casas, 2005). The selected security measures must respect these limitations.

Three concerns are within the scope of this thesis: low computation overhead, low communication overhead, and data confidentiality. The computation overhead directly impacts the mote's battery lifetime. A low communication overhead helps reduce power consumption due to antenna usage as well as preserving bandwidth. Data confidentiality ensures the content of a message is only understood by the intended recipient.

Many researchers have addressed aspects of WSN security. As discussed below, the TinySec library addresses the message integrity and confidentiality within a ZigBee network. SPINS focuses on preventing replay attacks and on providing broadcast message authentication. LEAP enables multiple key distribution mechanisms for different messaging scenarios (e.g. The Network Coordinator sending a message to a specific mote). Sizzle supports Internet SSL access through a specialized gateway mote that uses the HTTP stack. The "resurrecting duckling", though not directly applied to WSN,

www.manaraa.com

defines a security policy to introduce an embedded system into a network, to ensure the device performs only authorized actions, and to permanently remove an embedded system from the network; several interesting off-shoots of this research have been developed.

2.4.1 TinySec

UC Berkeley designed TinySec (Karlof, Sastry, Wagner, 2004) to achieve access control, message integrity, and message confidentiality within WSN. TinySec uses message authentication code (MAC) to authenticate message, initialization vectors (IV) to mask packet contents, and the Skipjack algorithm (Schneier, 1996) to encrypt messages.

TinySec implements MAC to achieve message authentication. A MAC is a one-way hash function paired with a key. The hash is appended to an encrypted message and used as a checksum. A mote with the same hash key may recalculate the checksum to verify the authenticity of the packet data (Schneier, 1996).

Intialization vectors disguise packet content within TinySec. WSN messages may be highly repetitive such as when motes respond with a simple YES or NO packet. Eavesdroppers may study the network messages and behavior of a network and learn to distinguish YES packets despite encryption. An IV forces repeated plaintext messages to become unique by appending a variable block of unique data to the packet. The simple message, “YES,” frequently sent by a fictional network would be changed to, “123YES,” and later when repeated changes to, “964YES” (Schneier, 1996).

The Skipjack encryption algorithm was selected for its conservative RAM usage and processing speed. UC Berkley researchers rejected AES (Schneier, 1996) and Triple-

www.manaraa.com

DES (Schneier, 1996) as being too processor intensive and too slow (Karlof, Sastry, Wagner, 2004).

TinySec is flawed as it does not prevent a message from being replayed by an unauthorized device at a later date (a quality known as data freshness). The RAM-constraints of a typical mote prevent the creation and maintenance of a table with the last IV value sent from each of the other mote on the network. Further, TinySec may not be practical as it is built upon a mote Operating System, Tiny OS, and the additional computation and footprint required to include the OS on a mote may exceed the abilities of some motes.

2.4.2 SPINS

SPINS (Perrig et al, 2002), developed by Carnegie Mellon University, emphasizes data freshness over confidentiality. SPINS is developed on the assumption that constrained computation abilities of motes prevents the use of public keys. Two technologies create the foundation of SPINS: SNEP and μTESLA.

The SNEP portion ensures data freshness. Each mote knows a pre-distributed master key common among all network motes, and each mote has a counter. To establish communication, the client mote sends its current counter number to the mote it wishes to contact. The server mote returns its own counter to the client with the MAC calculated using the server's calculated MAC hash. The client then responds with its own counter and MAC calculated using its own MAC hash. Following this exchange, motes continue to communicate using a calculated encryption key, a pseudorandom block of code, and MAC hashed with a calculated key. All keys are based on the current mote counter.

www.manaraa.com

The µTELSA technology – adapted from a stream authentication called TESLA – provides broadcast authentication. Essentially, each key is self-authenticating as it is based on a key previously sent to the receiving mote; the initial parameters are distributed unicast by the network base station (Liu, Ning, 2007). The broadcasting mote repeatedly hashes a key over a period of time. Each message is appended with a MAC using a hash of the previous message's key. Each receiving mote knows the duration of time between messages and also calculates the chained MAC key to authenticate each message (Luk, Perrig, Whillock, 2006).

2.4.3 LEAP

LEAP (Localization Encryption and Authentication Protocol)(Zhu, Setia, Jajodia, 2003), designed by George Mason University, supports multiple keying mechanisms with which to encrypt four classes of mote data exchange: controller to mote, mote to mote, mote to multiple neighboring motes, and mote to all motes. Additionally, µTESLA is used for authenticated broadcasts by the controller mote. Periodic group rekeying prevents compromised motes from decrypting group messages with the current key.

The network controller assigns a unique key to every mote within the network. The network controller contains a master key known only to the controller mote. A unique key is generated with a pseudorandom function using the mote's unique id and the controller mote's master key.

Each mote shares a pairwise key with each neighbor mote within one hop. During the initial introduction of a mote to the system, the controller mote provides the mote with an initial, generated key. The mote then generates its own key using the initial key as a seed. When the mote is deployed, it discovers it's near neighbors and exchanges the

www.manaraa.com

mote-generated key. The mote then uses its own key and its neighbor key to generate a new, shared key and deletes the key originally given by its neighbor. The mote repeats this process for each of its neighbors. Thus every mote shares a unique key with each of its neighbors. If a mote is added to the network after the initial set up, it may obtain a list of neighborhood motes from an established neighbor.

Clusters of motes may be established. The mote initiating the cluster generates a random key, encrypts the key with the pairwise key for each neighbor, and sends the cluster key. The receiving mote decrypts the message to obtain the cluster key. The receiving mote may encrypt this key and share it with its own neighbors.

Finally, a mote may generate a pairwise key specific to a mote several hops away. By network discovery a mote may indentify motes further than one-hop away. Using a process similar to the establishment of pairwise keys with one-hop neighbors, a pairwise key may be established with n-hop motes. This scenario is useful during data aggregation and allows the data to only be decrypted by the end mote.

A great fault of LEAP is its assumption that all motes are static. Even more troublesome, the establishment of a unique, pairwise key with every network mote a sending mote may wish to communicate with can easily become an onerous, RAM-consuming task.

2.4.4 Sizzle

Sun Microsystems developed Sizzle to create a secure Internet connection to a wireless sensor network (Gupta et al, 2005). Their abbreviated SSL protocol and small-footprint HTTPS stack require less than 4KB RAM. A single mote acts as a gateway between an Internet connection and the remainder of the WSN. Sizzle successfully

www.manaraa.com

addresses Internet access to a wireless sensor network including establishing a secure connection and transporting encrypted data. Sizzle does not address communication between motes.

2.4.5 MiniSec

In 2007, some of the same researchers who developed SPINS at CMU introduced a new WSN security architecture named MiniSec (Luk et al, 2007). MiniSec focuses on network layer security and assumes pre-established symmetric keys. This architecture seeks to provide a secure network layer protocol providing a high level of security with low energy consumption. These goals are achieved by operating through a block cipher mode, transmitting only a portion of the IV (unlike TinySec and SNEP), and creating a distinct protocol for broadcast messages and unicast messages.

2.4.6 Resurrecting Duckling

The resurrecting duckling (Stajano, 2002; Stajano, Anderson, 2002) is an embedded system security policy designed to address the issues of device authorization to command actions and perform actions. The basis of Stajano's policy is comparable to the hatching of a duckling – the duckling imprints itself to the first being it sees as its mother. A device is "soulless" until it is introduced to its mother who provides it with a soul. The authors recommend assuring the validity of the soul by transmission only via IrDA or a physical link. This "soul" is a policy dictating what actions the duckling may perform: the exchange of information, the adherence to commands, and the issuing of commands. This policy also dictates that a device can "die," or cease to use its policy; thereby returning to a state without imprint. After the imprinted policy has been removed from a

www.manaraa.com

device it may be introduced to a new mother device. This policy is suitable for a WSN in which motes must be recognized as a network member despite frequently entering and leaving the network and a WSN in which motes may participate in group relationships.

Several Zigbee researchers at CMU have taken note of Stajano's work (Kuo, et al. 2007). The goal of the researchers' "Message-in-a-bottle" is to provide a secure means to initialize a mote cryptographic key without requiring specialized hardware. They observe that the resurrecting duckling may require specialized hardware in order to securely transmit the initial key. The "bottle" of their solution is a faraday cage that protects the key transmissions from eavesdroppers and frequency jamming.

2.5 Related Work

Numerous research and hobbyist projects exist using 802.15.4-compatiable hardware kits. Unfortunately, it is extremely rare for the corresponding publications to provide details on the decision process, rationale, or experience in using their selected hardware. There are only two publications detailing the selection of a development kit.

The first publication is a paper presented at an ASEE (American Society of Engineering Educators) conference regarding a hands-on introduction to ZigBee for undergraduates co-written by the author of this thesis (Bateman, Armstrong, and Helps, 2007). This paper only mentions the qualities desired in the development hardware and notes which kits were considered. Few details regarding the selection of one development kit of another are provided by the authors, and the topic of security is never addressed.

The second publication is a master's thesis examining the suitability of ZigBee for wireless applications (Andersson, 2007). In his thesis, Andreas Andersson provides a

www.manaraa.com

brief discussion on the market's opinion of several development 802.15.4-compliant transceivers. He notes that the CC2420 receiver is the most commonly used within hardware kits (including Ember, Microchip and Crossbow). He also notes that the ZigBee development community infrequently uses the Freescale MC13192 that may indicate that they are impressed by neither the chip's price nor its performance. Andersson further compares 7 development kits based on the (then) availability of a ZigBee-compliant stack, the number of nodes in a kit, the transceiver used, the size of the company developing the kit, and the cost of the kit. Ultimately, he chose the CC2320 DK from Chipcon because it offered a more complete, ZigBee-compliant stack, the best transceiver, and the shortest time for delivery.

2.6 Conclusions

IEEE 802.15.4 and its derivative ZigBee are the standards best suited for WSN communication, yet they each rely on a developer to ensure the security of the channel between two devices. Much research remains on security techniques with WSN, and the impact of the security techniques on the limited resource environment needs to be studied. At this time, very little has been published regarding the selection of a development kit emphasizing the purpose of security research.

www.manaraa.com

3 Methodology

In this chapter the methodology necessary to address the problem stated in Chapter 1 is dicussed. Several development kits are selected for evaluation. A security scenario to measure the platforms is presented. The scenario is the assessed for the qualities necessary in a development kit. Finally, additional development requirements and concerns are discussed.

3.1 Candidate Development Kits

The objective of this thesis is to compare several, readily available hardware kits for WSN security development. The selection process for the demonstration hardware shall be completed in four phases. First, the hardware characteristics necessary to compare the development kits shall be discussed. Second, the hardware characteristics eliminated from consideration shall be noted. Third, the notable hardware development kits not selected shall be mentioned. Fourth, the hardware development kits to be evaluated shall be presented.

3.1.1 Necessary Hardware Characteristics

Several characteristics must be considered when assessing the candidate development kits: MCU (micro controller) and memory resources, user interfaces,

www.manaraa.com

integrated sensors, power supplies, and technical support. These five characteristics influence the ease of development, and they may limit the design of the proposed solution.

Insufficient processor and memory resources may restrict which security tools are viable on the development hardware. The available RAM defines the number of private keys a mote may store; thus, the capability of the hypothetical network may be impeded if a security policy requires that a mote record the private key of each networked mote. Similarly, the memory storage space and processor size may be insufficient to run cryptographic code or may exclude the use of resource intensive algorithms such as RSA.

Alternative developmental interfaces influence the ease of providing feedback to the code developer. A USB or serial link may provide confirmation of code execution and program traces to a host computer. An LCD on a mote may display clear messages without the aid of a host computer. LEDs may also communicate data to the developer.

Integrated sensor capabilities influence the system application and affect a programmer's debugging strategies. Common integrated- and daughterboard-sensors include general I/O ADC and DAC, push-buttons, thermistors, phototransistors, and accelerometers. On one hand, this thesis allows the demonstration application to be designed around the available mote sensors. On the other hand, the sensors of the evaluated hardware are highly relevant because sensors provide a means of debugging and application input. For example, the inclusion of a single push button allows the developer to easily control the mote in order to simulate input for debugging; whereas a lack of any push buttons and an inclusion of a photodiode require the developer to control the mote's environment (e.g. a rigged shoe box) in order to simulate input for debugging.

www.manaraa.com

The cost of research may be influenced by the cost of a power supply to run the hardware. Development motes are designed to operate with less optimization, fewer sleep cycles, and increased processing power requirements compared to a completed, manufactured mote. The resulting higher rate of battery consumption may require a researcher to frequently replace a mote's batteries or invest in rechargeable batteries. Development motes are often powered by one or more of the following: AA batteries, USB cables, and 110 V AC/DC converters.

While not a physical hardware component, another consideration is the candidate hardware's available technical support. The quality of documentation, technical support, and demonstration code may ease the development process for the unfamiliar hardware. The presence of an online developers' community may assist in troubleshooting hardware difficulties and software bugs. Completed, prior research successes using the mote hardware demonstrates the implementation feasibility of the hardware.

3.1.2 Disregarded Hardware Characteristics

The three characteristics excluded from consideration include the hardware programming language, mote physical size, and mote power consumption. The programming language is irrelevant to the selection process because hardware kits commonly run C/C++. (The only known exception is the Sun SPOT which runs Java.) The physical size and power consumption of the development motes will not be considered because these optimizations are unnecessary for this research.

www.manaraa.com

3.1.3 Notable Excluded Platforms

Several kits common to embedded system development have been excluded from consideration. The Gumstix lacks a readily available 802.15.4-compatible stack. The PICDEM Z, despite several years on the market, has not garnered any market support. Finally, TelosB was disregarded in favor of its sibling product, MICA Z, which is ZigBee-compliant. The Ember EM250 is seriously considered because of Ember's popularity in industry; however, this device was rejected due to expense of both its kit and its expansion boards. The TI CC2430 and TI CC2431 and are also considered, yet each are disregarded due to the expense of the kits.

The Gumstix (Gumstix Inc., 2009) is a small motherboard – literally about the size of a gum stick. This device has been used in many applications: from creating a clarinet-playing robot (Wolfe, 2008) to gathering and viewing real-time measurements of probes in the San Francisco Bay (Google, 2009). Bluetooth and Wi-Fi capabilities may be added to the Gumstix with a Wi-Fi module. In 2007, Research Studios of Austria noted on their website that they had completed a Gumstix-ZigBee adaptor, but no further information, publications, or software releases could be found.

The PICDEM Z (Microchip Technology Inc., 2008) is a PIC18 motherboard and daughterboard kit for use with the MPLAB IDE. This kit was excluded because its only 802.15.4-compatible stack was a vastly trimmed version of ZigBee. Some researchers experienced with Microchip embedded systems may find this kit favorable because it is compatible with the MPLab IDE, ICD2, and MCC18 compiler. The only notable PICDEM Z published research evaluates the signal strength of wireless sensor networks within indoor scenarios (Ferrari, 2007).

www.manaraa.com

The TelosB is an 802.15.4-compatible, Crossbow product (Crossbow Technology Inc., 2008b). The device was originally design in conjunction with WSN research at UC Berkeley, so the platform depends on TinyOS. Examples of research include a Purdue study on how cryptography effects node lifetime (Piotrowski, Langendoefer, and Peter, 2006) and creating a mesh network to transmit audio and video (Song, Hatzinakos, Wang, 2008). It was not considered as a hardware kit in favor of Crossbow's ZigBee-compliant MICAz.

Ember sells a ZigBee-compliant device called the EM250 as part of their InSight ZigBee development kit. The Em250 data sheet (Ember Corporation, 2006) indicates that the device is built with a 16-bit XAP2b microprocessor, 128 KB program memory, and 5k of RAM. No information is provided regarding additional communication hardware (i.e. Serial or USB). Ember offers very detailed documentation on the EM250 microprocessor; however, they provide little information about the mote hardware itself. This device was rejected for this research because, at the time of writing, the InSight development kits range from thousands to tens of thousands of dollars (Digi-Key Corporation, 2009).

Chipcon, the creators of the devices used in Andersson's research, was acquired by Texas Instruments in 2006 (Mumford). Their most recent ZigBee development platform is the CC2430. Of all the available hardware kits, TI appears to offer the most complete kit: a ZigBee-compliant stack, 2 evaluation boards 2 evaluation modules, 2 demonstration boards, batteries, a C-compiler with debugger on a 90-day evaluation license, and a copy of the Daintree sensor network analyzer (later described in Section

www.manaraa.com

4.8). Unfortunately, this more complete kit comes at a cost (starting at $1,500), so is will not be evaluated within this thesis.

3.1.4 Hardware for Evaluation

Many companies offer development hardware and microchips for 802.15.4 WSN applications. Both ZigBee-compliant and 802.15.4-compliant kits are considered – though more weight it given towards ZigBee-compliant kits due to the specification's more thorough security definition. The hardware considered – those which were readily available at the time of hardware selection – include the Crossbow MICAz, the Freescale MC1321x, and the Sun SPOT.

In general WSN literature, the word 'mote' frequently refers to a node within a wireless sensor network. It should be noted that the Crossbow claims the word 'mote' to describe any of its WSN node products. Sun Microsystems literature prefers to call their device 'SPOT' and refer to Crossbow's device as 'mote.' Throughout the Freescale documentation they use the unwieldy label 'MC1321x' or 'ZigBee End Device' (ZED). This thesis shall use the words 'mote,' 'node,' and 'device' interchangeably regardless of company origin. MICAz, SPOT, and MC1321x shall be used to directly refer to the given product.

The Crossbow, Freescale, and Sun kits all have sufficient resources for the research outlined here, and are affordable enough that they are accessible. A more thorough examination of their specific qualities is included in Chapter 4.

www.manaraa.com

3.2 Security Scenario

A security development scenario will be used to determine which software qualities are most important in a study of WSN key distribution and updating. This security development scenario will address a few common issues in creating a security policy, and partially addresses a key management policy. The hypothetical security policy shall define means to establish the user's approval of a node within the network, to create a session between two network motes, to periodically update the network key, to request the current network key, and to remove a mote from the network. The security scenario assumes that the network tasks a Network Coordinator with the management of the network security.

3.2.1 Description of the Solution

This section expounds on the security scenario. The primary objective of this policy is to establish a secure exchange of data between motes within a WSN network. The tasks critical to the success of the scenario include the acceptance of a new node into the network, the distribution of network keys, and the establishment of secure sessions.

The Network Coordinator shall control the security and authorization of its network. When a mote wishes to join the network, it must first authenticate with the Network Coordinator. If the mote is permitted to join the network, the Network Coordinator will provide the current network key to the joining mote (this transaction is detailed in Section 0). Thereafter, the Network Coordinator will generate and distribute a new key at the end of a defined time; if a mote misses this update it must contact the Network Coordinator to receive the new key.

www.manaraa.com

Communication between motes shall be secured by a session key. The handshaking necessary to establish the session key shall be based on SSL (Secure Socket Layers) and all handshaking packets shall be encrypted with the network key. Motes are encouraged to formally close the session with a termination packet, but a timeout scenario will prevent system deadlock. A session may not be reestablished once it is terminated.

Symmetric cryptography was selected to secure the communication channels of the proposed security scenario. The network master key is used to encrypt and secure messages and the same key is used to decrypt those messages. This prevents any device without the master key from decrypting the packets. Symmetric keys were selected because it is a well-studied cryptographic technique that can be deployed on limited resource devices. Asymmetric key cryptography was not used due to concern that motes would have insufficient RAM to store a key for each device in the network. Similarly, digital certificates are excluded from the proposed security policy due to anticipated RAM constraints. There are several known weaknesses of symmetric cryptography – namely the initial key distribution. In a sense the Network Coordinator functions as the key distribution center (KDC), but not to the extent that a mote may contact the KDC to verify the network key. Another known weakness is the distribution of the initial key – how does one securely transmit a key when two devices do not yet share a secure communications channel or a secret key by which to encrypt and decrypt? Both 802.15.4 and ZigBee rely on a pre-installed key placed on the hardware either by coding or configuration. The proposed security policy transmits this key in clear text; a large vulnerability. This could be worked around by requiring the initial join process to be

www.manaraa.com

completed over a secure wire or could be secured by locking each device in a faraday cage before transmitting the initial key wirelessly. Neither solution is completely satisfactory, but the issue at hand is appropriate for a separate study in WSN security and is outside the scope of this thesis. This thesis will accept the symmetric key limitation of 802.15.4 and improve the security of it by proposing temporal key updating policy.

3.2.2 Network Entities

A typical network topology will consist of one Network Coordinator and one or more Network Motes. This is consistent with the ZigBee specification designation of a specialized node charged with managing the network security of a network of typical nodes.

3.2.2.1 Network Coordinator

The Network Coordinator shall be responsible for initiating the network, verifying motes wishing to enter the network, distributing the network key, and maintaining a list of all mote groups. The Network Coordinator shall have the following properties.

1. The Network Coordinator shall admit new motes into the network.
2. The Network Coordinator shall generate the network key.
3. The Network Coordinator shall generate a new network key after a defined period of time.
4. The Network Coordinator shall store one prior network key.
5. The Network Coordinator shall distribute the network key to new motes admitted to the network.

www.manaraa.com

6. The Network Coordinator shall provide the current network key to valid motes which know the prior network key.
7. The Network Coordinator may update the new network key by sending a message to each mote within a session established with the prior key.

3.2.2.2 Network Mote

The term "Network Mote" shall refer to any device except the Network Coordinator. Each Network Mote shall maintain its functions as defined by 802.15.4 specification (i.e. routing). Each Network Mote shall have the following properties.

1. Motes shall connect to the network prior to being authenticated by the Network Coordinator.
2. After joining the Network Coordinator, the mote shall complete the authentication process.
3. Network Motes shall store only one network key.
4. Should the Network Mote miss a periodic network key update from the Network Coordinator, the Network Mote shall request the new network key.
5. If a Network Mote misses more than one network key update, the Network Mote must rejoin the network.
6. Whenever possible a mote shall inform the Network Coordinator when the mote is permanently leaving the network.

www.manaraa.com

3.2.3 Joining a Network

Every mote must be introduced to the network. This initial communication shall occur over a wireless link in clear text. For greater security the initial communication may be transmitted via a hardware link, but this thesis shall not implement that option.

Like the Resurrecting Duckling policy, a Network Mote shall be loyal first and foremost to the Network Coordinator. The physical press of a mote button and the physical press of a coordinator button shall validate the user's intent to include the mote in the system network. Once the mote is validated, the Network Coordinator shall provide the mote with the network key (i.e. "imprinting").

The hypothetical security policy resembles the security mechanism of the ZigBee-2006 specification with two critical additions. Both policies require the existence of one and only one mote trusted with the security of the network. Both policies require that trust center (i.e. the base station) manage with the distribution of a network key. The proposed policy, however, adds the requirement of a physical permission to transmit the key, allows the Network Coordinator to generate a network key rather than relying on a pre-loaded key, and establishes the requirement and processes that the network key be periodically updated.

The Network Join Communication sequence is illustrated in Figure 3-1. This diagram emphasizes that the mote must join the network (i.e. the nebulous, ether, communication cloud) before messages of any sort can be transmitted or received. All transactions are completed in clear text. First, the mote joining the network sends a packet to the Network Coordinator requesting the initiation of the handshaking. The Network Coordinator responds with an ACK. The joining mote and the Network

www.manaraa.com

Coordinator then each request the user to verify the join with either a keystroke from the host computer or a press of a button on the mote. If the user responds within the allotted time (e.g. 10 seconds), the Mote then sends an ACK to the Network Coordinator. Likewise, if the user responds within the allotted time and an ACK is received from the joining mote, the Network Coordinator records the mote's entry into the network. Then the Network Coordinator sends the network key to the mote within the CoordinatorWelcome packet. Finally, the mote then saves these settings.

The activity of the Network Coordinator is shown in Figure 3-2 and the activity of the Mote joining the network is shown in Figure 3-3. These diagrams each show the program flow of each device as it transmits the packets of the Network Join communication sequence. These diagrams emphasize how each device avoids deadlock by requiring that the corresponding device respond within an allotted window of time. When the Network Join sequence is unsuccessful (i.e. an error occurs) than the mote still wishing to join the network must restart the entire Network Join process. Upon successful completion the mote is ready to begin transferring data with any member of the system network.

www.manaraa.com

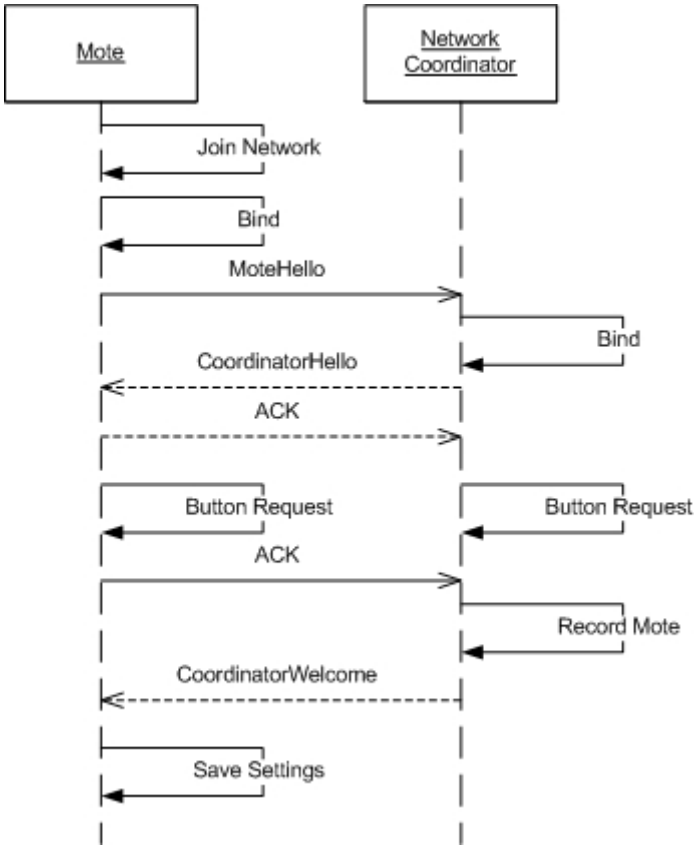

Figure 3-1: Network Join Communication Sequence

www.manaraa.com

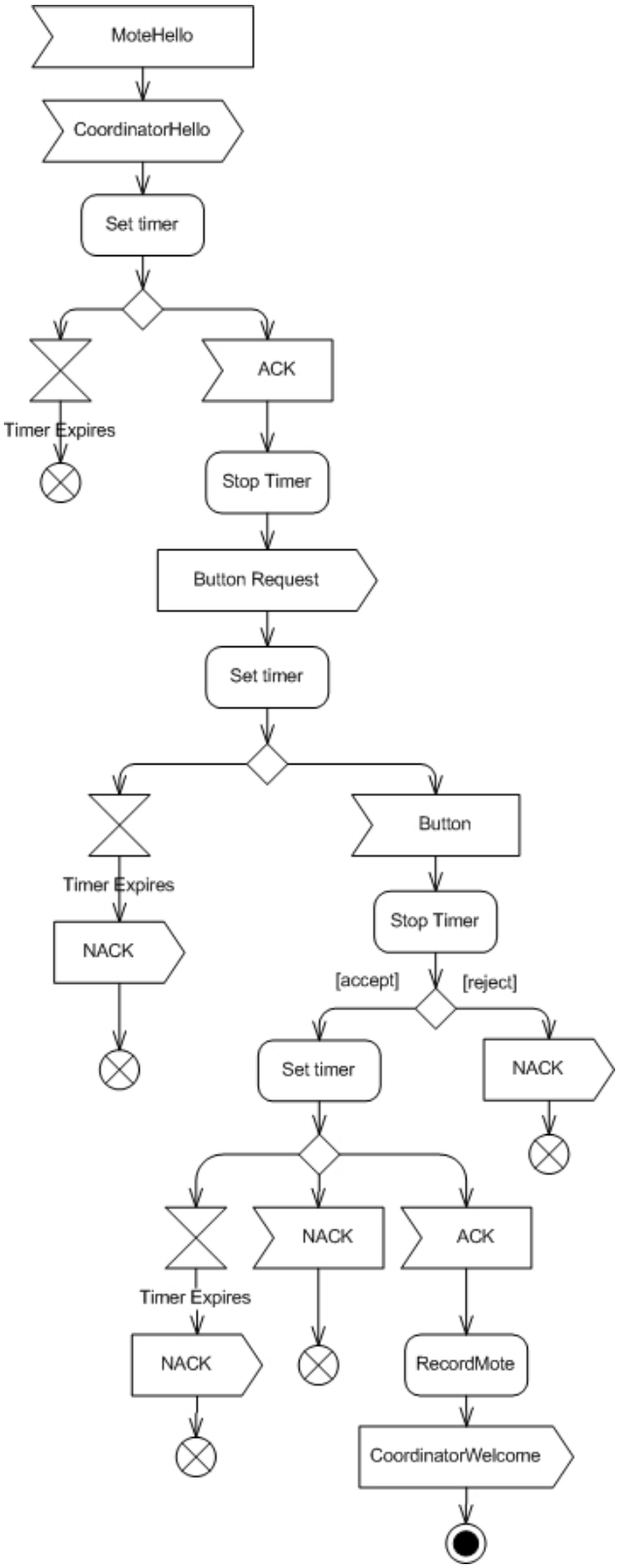

Figure 3-2: Network Join Network Coordinator Activity

www.manaraa.com

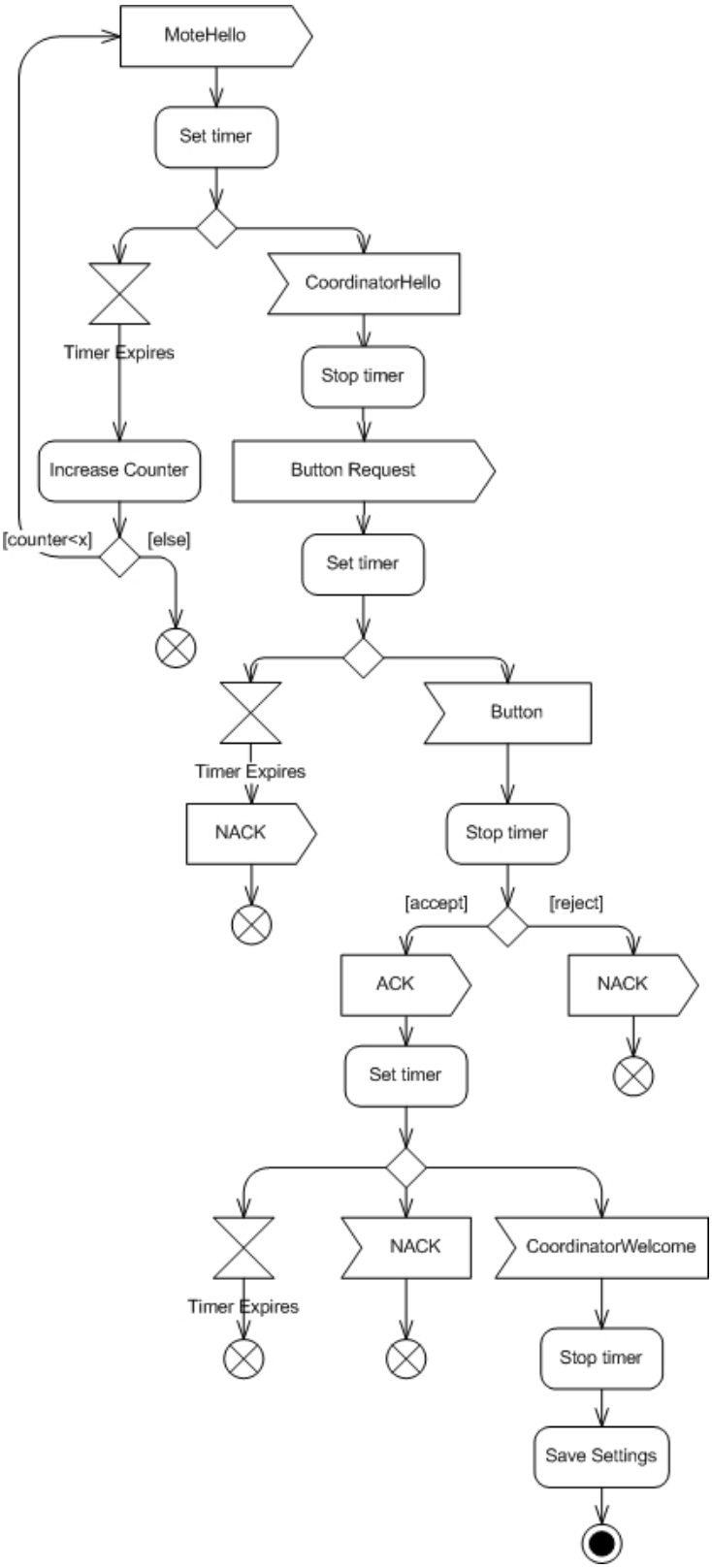

Figure 3-3: Network Join Mote Activity

www.manaraa.com

3.2.4 Establishing a Session

Any two motes with the current network key may exchange data after establishing a session. Secure sessions shall be established using SSL (Secure Socket Layers), abbreviated to accommodate small processors and limited bandwidth. The SSL security protocol is commonplace among client-server transactions; however, SSL is designed for systems with which have faster processing and have better reliability than the typical WSN. The number of necessary SSL packets to establish communication shall be reduced by eliminating certificates and by limiting each mote to a single encryption method. This also reduces required code overhead, RAM usage, bandwidth, etc.

The network key may periodically be updated (described in Sections 3.2.5 and 3.2.6). Any mote as the client with the current or prior network key may establish a session with the Network Coordinator as the server; any mote-client not using the immediate last key or the current key shall be rejected. Motes must have the current network key when communicating with other motes.

Within the Session Establish sequence is the client is charged with generating a random, session key. This session key is included in the ClientHello packet which is encrypted with the network key and transmitted to the server. (The initial transmission of the network key is described in Section 3.2.3.) All Session Establish packets shall be encrypted by the network key until the ChangeCipher command is issued at which point all communication will commence encrypting with the session key. Once a session is established, all communication shall be encrypted using the session key until the session is terminated.

www.manaraa.com

To prevent system deadlock, the client shall terminate each session with a CipherFinished packet, and either the client or the server will timeout if after a given period of time the data transfer does not commence. Forbidding the motes from storing session keys will conserve mote RAM, thus a session may not be resumed once it is terminated.

3.2.5 Updating the Network Key

To adhere to best practices, the Network Coordinator shall update the network key at the end of a defined time period. This key update transaction assumes all motes are present on the network at the time the update packets are transmitted. For simplicity, this scenario intentionally omits addressing the issue occurring when a mote is not present on the network during a key update transaction. (Section 3.2.6 addresses how a mote can return to a network after having missed a key update.)

The packet sequence of distributing the new network key is illustrated in Figure 3-4. First, the Network Coordinator overwrites the prior key with the current network key, generates a new key, and stores the new key as the current network key. Then the Network Coordinator establishes a session using the prior key and sends the new network key to each mote one by one.

www.manaraa.com

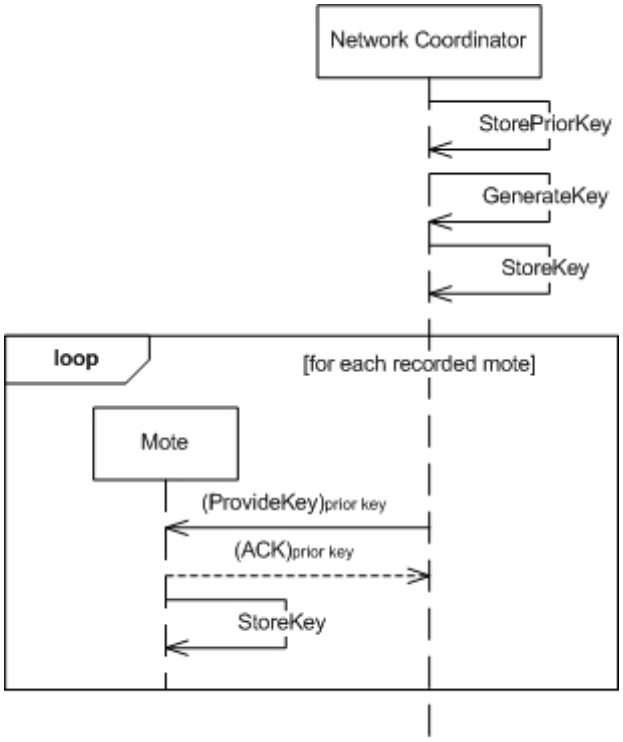

Figure 3-4: Key Update Communication Sequence

www.manaraa.com

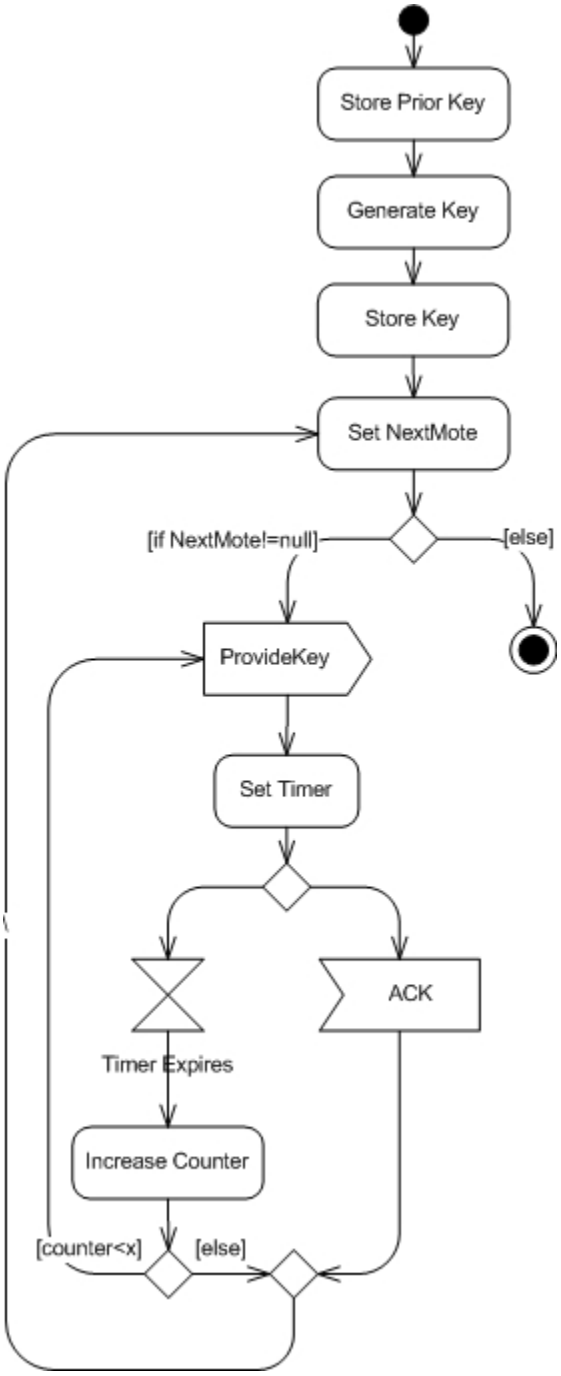

Figure 3-5: Key Update Coordinator Decision Process

www.manaraa.com

The decision activity of the Network Coordinator is shown in Figure 3-5. If a mote does not acknowledge the receipt of a packet within a user-defined number of attempts than the Network Coordinator proceeds to contact the next known mote – no record of failed key distribution is kept. The key update process is considered complete after the Network Coordinator has attempted to distribute the new key to each mote.

The decision activity of the Network Mote as shown in Figure 3-6 is simple. When a mote receives a new network key packet it acknowledges the receipt of the new key, and overwrites its existing network key.

Figure 3-6: Key Update Mote Decision Process

3.2.6 Requesting the Network Key

A mote with an expired network key cannot communicate with other networked motes which have the updated network key. The Key Request process exists to allow motes absent from the network during the Network Key Update process (described in Section 3.2.5 to obtain the new key.

When a mote's network key has expired the mote must request the current key from the Network Coordinator. Figure 3-7 shows the sequence of packets exchanged

www.manaraa.com

when an authorized mote requests the new network key from the Network Coordinator. During the data exchange portion of a session established with the Network Coordinator using the prior network key the mote requests the new key. Then the Network Coordinator responds by providing the new network key. Finally, the network mote stores the provided key as the new network key.

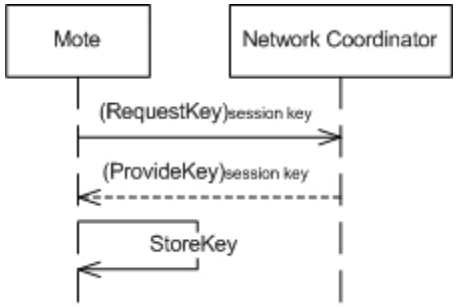

Figure 3-7 - Key Request Communication Sequence

The Network Coordinator is charged with responding to mote requests for the new key (shown in Figure 3-8). Any mote with the prior network key may establish a secure session with the Network Coordinator; however only authorized motes shall receive the updated network key. A mote will be refused the new network key if the Network Coordinator cannot verify the mote as part of the system, or if the mote has missed more than one network key update, or if the mote has been removed from the system. During a session date exchange, the mote may attempt to request the new network key (shown in Figure 3-9).

www.manaraa.com

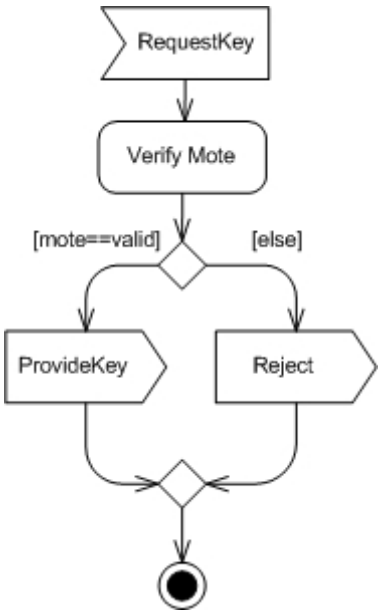

Figure 3-8- Key Request Coordinator Decision

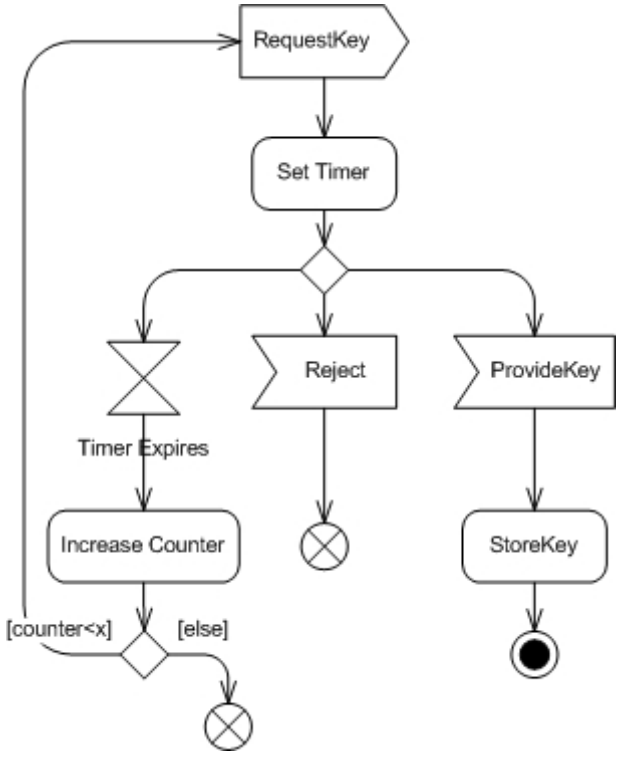

Figure 3-9 - Key Request Mote Decision

www.manaraa.com

To enable motes to request the most recent network key using the prior – and only the prior – network key, the Network Coordinator may be required to attempt to decrypt a packet with either key. Figure 3-10 illustrates the decision process the Network Coordinator shall follow when determining which key to decrypt with. Essentially, it is a trial-and-error approach. The correct key must have been selected if the packet can be successfully decrypted and restored to a legible cleartext packet (i.e. ClientHello).

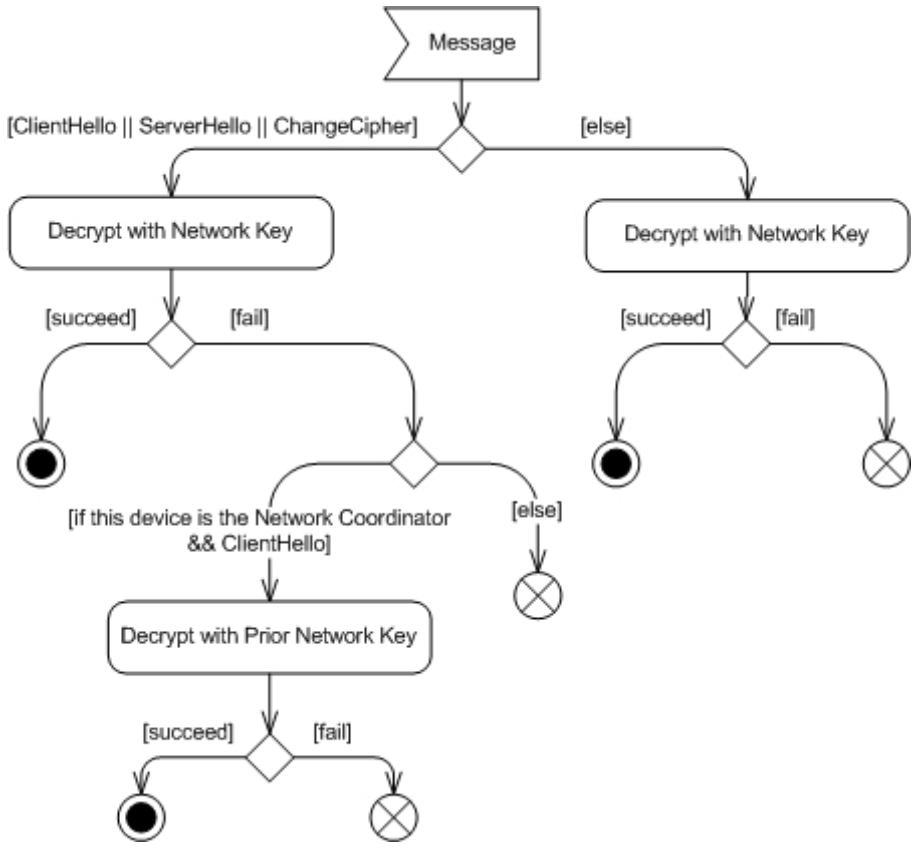

Figure 3-10 – Network Coordinator Decryption Decision

Should the decryption process fail the Network Coordinator shall not respond to the out-of-date Network Mote, and the Network Mote shall treat the lack of response as a no-response, timeout.

www.manaraa.com

3.2.7 Exiting the Network

Motes may remain a member of the network indefinitely until it is removed in one of three manners. First, a mote will automatically be refused by the network if a mote misses two or more network updates; that is, when a mote does not have the current network key and when the Network Coordinator has expired the prior key. Second, a user may initiate the exit through the mote application which will then erase the mote's stored keys and inform the Network Coordinator that the mote is no longer a network node. Third, a user may remove the mote through the Network Coordinator application which will then instruct the mote to erase its stored keys (admittedly, only obedient motes will comply with this command) and exclude the mote from the network key updates. After a mote has been permanently removed from the network the mote cannot communicate with the system without completing the Network Join process. The ZigBee-2007 specification provides mechanisms for device management that may be useful to implementing this hypothetical security policy transaction.

3.3 Software Components

The vital resources necessary to develop and test the security policy are revealed by observation. These resources include editable transfer protocol, timers, and an existing encryption algorithm. A development platform must include each of these resources to enable the efficient development of an effective security policy. For example, a researcher exploring how to best implement session handshaking should have access to a provided encryption algorithm that will fit within the device's flash memory rather than requiring the researcher to find or develop a suitable encryption algorithm.

www.manaraa.com

It is necessary that a transfer protocol exist that can be modified and extended by the researcher. A developer ought to be able to access and set the values of the existing 802.15.4 or ZigBee headers. Several of these header settings may use useful such as the key identification fields or the security type fields. In some instances it may be necessary to expand the headers without violating the standard's constructs.

Timers are critical to measuring time intervals. These intervals may be used to calculate when to next release the periodic key update, and to time the communication between two devices to prevent system deadlock. Timers may also be used when developing tests to ensure the quality of a proposed solution (i.e. measuring the increased time necessary for a mote to join the network).

An existing encryption algorithm that can fit with the development kit's available flash and RAM is crucial. A sufficiently cryptographically secure pseudo-random number generator is also crucial. The security of the test scenario will fail if its components are not reliably secure.

3.4 Key Development Concerns

In Chapter 4, each development kit will be evaluated for its suitability for academic research in security. The hardware topics discussed shall include the sensors, displays, communication interfaces, power supply, device types, daughter boards, housing, and cost. The software topics shall include the available stack, demonstration code, and any additional security libraries. The development tools section shall include notes on the easy of loading, compiling, debugging, cost, and overall usability of the development applications. The development assistance for each kit shall be evaluated

www.manaraa.com

including available manuals, tutorials, data sheets, and online community. Finally, other development resources such as simulators and benchmarking tools shall be noted for each kit.

www.manaraa.com

4 Analysis

This chapter shall examine and compare the Crossbow MICAz, the Freescale MC1321x, and the Sun SPOT. The background and basic characteristics of each kit shall be introduced. The hardware shall be evaluated based on sensors, user interface, daughter boards, power options, housing, and the multiple device types. The software shall be assessed based on its language, radio stack, demonstration code, additional libraries, Java Virtual Machine (JVM), and available operating systems. The hypothetical security scenario will be analyzed and each platform shall be audited for the security resources required by the scenario. Remarks shall be made regarding the hardware's available documentation, demonstration code, and online community. Additional simulation, emulation, and benchmarking tools available to the devices shall be noted. The expense of the hardware and software of each kit will be compared. Finally, the evaluation of each hardware kit shall be summarized.

4.1 Introduction of Evaluated Hardware

Three readily available hardware kits were selected for evaluation in Chapter 2: Crossbow MICAz, Freescale MC1321x, and Sun SPOT. They were each found to have the basic hardware characteristics necessary to serve as a platform for WSN security research. A basic background and summary of each device shall now be presented.

www.manaraa.com

4.1.1.1 Crossbow MICAz

The Crossbow MICAz has been selected for evaluation. This primary code base of this device is TinyOS (Levis, et al, 2005). This device fulfills all the established requirements, offers more flash memory than its competitors, and finds popularity within WSN research communities.

Crossbow Technology Inc. leads the industry in the number of WSN products offered to the research market. Currently, Crossbow sells three 802.15.4-compliant motes: MICAz, Telos B, and IRIS; of these products the MICAz was preferred for its ZigBee-compliance. The MICAz includes many desirable features including 128 KB of program storage, 4 KB of RAM, several LEDs, a variety of daughterboard sensors, and a 2 AA battery power supply (see Table 4-1) (Crossbow Technology Inc., 2007). Due to the mote's popularity among ZigBee researchers much MICAz documentation and many troubleshooting tips exist online. Researchers have used the MICAz to examine the possibility of implementing a WSN for medical sensor data (Hansen, Støa, 2006); to determine the feasibility of real-time, wearable, PAN devices (Koh, Kong, 2006); and to collect greenhouse environment data (Zhu, Zhong, Shir, 2006).

Table 4-1: Crossbow MICAz Features (Crossbow Technology Inc., 2007)

MCU	MPR2400 (based on the Atmel Atmega128L (16 MHz, 8-bit))
Program Memory	128 KB
RAM	4 KB
Communications	ZigBee, Serial UART
Power Options	2, AA Batteries
User Interface	3 LEDs
Size	58 mm x 32 mm x 14 mm
Sensors	51-pin Expansion Connector for Crossbow Sensor Boards (Light, Temperature, RH, Barometric Pressure, Acceleration/Seismic, Acoustic, Magnetic, Etc.)

www.manaraa.com

4.1.1.2 Freescale MC1321x

The Freescale MC1321x has been selected for evaluation. Freescale's inclusion of multiple LEDs, multiple push buttons, and an LCD on the network coordinator supplies multiple methods for data input and output. The serial port and the USB port provides alternative means of data transfer while troubleshooting the wireless connection. The mote's plastic casing allows for easy handling and protects the mote circuitry.

Freescale Semiconductors Inc. offers several different memory configurations of their MC1321x mote design. Each design may be purchased as an NCB (Network Coordinator Board) or SRB (Sensor Reference Board). The MC13213 SRB includes UART, USB, 60 KB Flash memory, 4KB RAM, 4 buttons, and 4 LEDs. Each mote is powered by either two AA batteries, a USB connection, or an 110V adapter (see Table 4-2)(Freescale Semiconductor Inc., 2007a). The NCB includes several additional hardware integrations such as an external antenna connection, an LCD, and an RS-232 port. Unfortunately, little research exists using these relatively new Freescale motes.

Table 4-2: Freescale MC13213 SRB Features (Freescale Semiconductor Inc., 2007a)

MCU	Freescale HCS08 (40 MHz, 8-bit)
Program Memory	60 KB
RAM	4 KB
Communications	ZigBee, USB
Power Options	2x, AA Batteries; USB; External Power Supply
User Interface	5 LEDs and 4 Switches
Size	86 mm x 56 mm x 30 mm
Sensors	Integrated 3-Axis Accelerometer; Temperature Sensor, 26-pin I/O Ports

Notably, Freescale now offers a MC1322x kit which was not available when the MC1321x was selected and purchased (Freescale Semiconductor Inc, 2008; Freescale

www.manaraa.com

Semiconductor Inc, 2009c). The MC1322x uses a 26MHz, 32-bit ARM processor, an AES hardware accelerator, 128 KB Flash and 96 KB RAM. Further, this device offers 80 KB ROM for the boot software. This kit also further specializes the Freescale ZigBee devices by creating a 'Low Power Board' (LPB) which runs on AAA batteries; the lower power profile appears to be achieved by removing the sensor board included in the SRB. Finally, the kit includes an 802.15.4 sniffer for monitoring nearby traffic.

4.1.1.3 Sun SPOT

As of March 2008, Sun has released all the code (including the JVM) as open source and introduced academic pricing available to students. Further, the SPOT offers the convenience of the Java language (this author's "native" programming language) and a small community of researchers implementing SSH on the Sun SPOT devices. Notably, the Sun SPOT is not ZigBee-compliant like the other two kits, but is 802.15.4-compliant.

Sun Microsystems Inc. offers a novel hardware design that runs a JVM directly on the hardware thus enabling Java development for the 802.15.4-compliant embedded system and eliminating the need for an Operating System. Manufacturing difficulties caused several release delays from the initially predicated release in 2006, but the hardware was eventually released in 2007.

Each Sun SPOT kit includes 1 base station and 2 full Sun SPOT devices. The base station acts as a gateway between a host computer and the WSN; the base station cannot run on battery-power and does not have any sensors. Each Sun SPOT device includes a 180 Mhz, 32-bit ARM processor, 512 RAM, 4 M flash, 8 tri-color LEDs, and 2 switches (see Table 4-3). The SPOT may be powered by a USB interface or a built-in lithium-ion battery The SPOT is significantly more powerful than the prior two evaluated kits, but

www.manaraa.com

there are no existing studies documenting how this affects the device's battery life in comparison to the other devices. The differences in specifications make it impossible for researchers to equally compare the Crossbow and Freescale motes to the SPOT where software efficiency and device lifetime are concerned. Empirical observation shows the Sun SPOT development motes to consume batteries more rapidly than others, but within the same order of magnitude.

Table 4-3: Sun SPOT Features (Sun Microsystems Inc., 2009a)

MCU	ARM920T (180 MHz, 32-bit)
Program Memory	4,000 KB (4 M)
RAM	512 KB
Communications	802.15.4
Power Options	Rechargeable Battery; USB Interface
User Interface	8, Tri-Color LEDs and 2 Switches
Size	41 mm x 23 mm x 70 mm
Sensors	Integrated 3-Axis Accelerometer; Temperature Sensor; Light Sensor; 6 Analog Inputs Readable by ADC; 6 General Purpose I/O Pins; 5 High Current Output Pins

Sun SPOTS have been implemented in several application-based projects such as art-installations (Sukumaran, 2006), collecting data in a Panama forest (Fusting, 2008), and guiding a robot with a gesture-remote (marvelouskobe, 2008). (It must be noted that there are very few periodical or academic publications featuring Sun SPOTs and many of the existing applications are from Sun Labs.)

An excellent feature of the Sun SPOT software is its open source code – including a security library. The library contains readily available code to generate ECC (Schneier, 1996) keys and perform SSL exchanges. The library expects the developer to create the application layer to manage the keys and establish the network interactions.

www.manaraa.com

4.2 Hardware

The hardware specification of each mote impacts the developer's ability to design software for the embedded system. The memory and processor capabilities may influence a researcher's freedom in using readily available algorithms. The user interface tools such as LEDs and push buttons will aid a developer in debugging mote software. The power options of a kit are notable as it may require a budget for batteries or require a mote to be tethered to a power supply; the power options also impact the motes usability in deployed experiments and applications. Good housing of a device is necessary to protect the circuitry of a device while not preventing the user from interacting with the device's interfaces. Crossbow, Freescale, and Sun each offers a kit with at least one standard mote that may be used as an endpoint or router within a WSN, as well as a specialized mote. Crossbow and Sun offer a mote for programming the network over the air, and Freescale offers a mote with an LCD interface and additional break-out pins.

4.2.1 Memory and Processor

In Table 4-1, Table 4-2, and Table 4-3 a comparison of each of the device hardware specs was presented. Each of the platform's memory and processer capabilities was found to be suitable for 802.15.4 and ZigBee development. On one hand, the MICAz and MC1321x are comparable. The Crossbow mote has 4KB RAM; 128KB Flash; and 16 MHz, 8-bit processor. The Freescale mote has 4 KB RAM; 60KB Flash; and a 40MHz, 8-bit processor. On the other hand, the Sun SPOT far exceeds the concept of "small, and limited" for a mote with 512 KB RAM, 4,000KB (4 M) Flash, and a 180 MHz, 32-bit processor. The Crossbow and Freescale devices will provide a more accurate test

www.manaraa.com

platform for real application expected to survive years on battery power alone, yet the Sun enables a security developer to use most common security algorithms without constraint.

4.2.2 User Interface and Sensors

The user interface of each mote is of great importance to the user. The interfaces of the device must enable the user to control the device actions, determine the point of the mote within a state diagram or process, debug the software code, and generate system output. Such interfaces may include a UART, LCDs, LEDs, and buttons. Sensor accessories such as accelerometers may be useful as well.

The MICAz offers the most limited interface of the evaluated devices. The mote itself only includes capabilities for processing and communication. An expansion connector is available for its daughterboards (discussed in Section 4.2.3). Only two LEDs are available on the device, and a third LED is reserved to indicate power. No buttons are available to the developer. The desktop may communicate and program the device either over the air (OTA) through a base station or via a serial connection.

Freescale MC1321x offers several interfaces and sensors. The mote offers 5 LEDs: 1 of which is dedicated to power and reset indication with, and the other 4 are available for use by the developer. The 4 development LEDS are situated above 4 buttons; though there is no hardware correlation between the lights and the buttons. Further, the NCB device offers an LCD for message output (see Section 4.2.6 for more information on Device Types). The device also has a 3-axis accelerometer, a temperature sensor, a 36-pin general purpose I/O, and an 8-channel, 10-bit ADC. The Freescale mote code is loaded from the development computer using a BDM (Background Debug Mode)

www.manaraa.com

interface; additionally, the BDM permits the user to debug and step through code while running it on the hardware.

The Sun SPOT provides several interfaces and sensors. The SPOT offers 8 tri-colored LEDS. These lights are a great benefit to programmers as it enables them to color-code messages. (For example, LED1 may be "blue" to indicate that no network connection has been established, "purple" to indicate that the mote is in the process of establishing a connection, and "red" to indicate a connection has been established.) A slight disadvantage is that the mote only offers 2 buttons – certainly an improvement over the MICAz, but not as convenient as the MC1321x. As with the Freescale mote, the SPOT offers a 3-axis accelerometer and a temperature sensor. The mote also offers a light sensor, 6 ADC pins, 6 general purpose I/O pins, and 5 high current output pins. Code may be loaded onto the SPOT via USB or by OTA distribution by a base station.

4.2.3 Daughterboards

Daughterboards may be attached to a mote to provide additional sensors or outputs. Neither Freescale nor Sun offer expansion daughterboards for their devices, yet each of those devices offer sufficient capabilities as they are. Crossbow, however, may cripple a developer with its lack of inputs and outputs. Fortunately, Crossbow offers a variety of daughterboards.

The MICAz has a built-in expansion connector. Sensing capabilities offered by these boards include GPS, magnetic field, sound, a light sensor, humidity, and temperature. These boards may not be useful for a developer except for very narrowly defined applications. The starter kit for the MICAz ships with 2 sensor nodes. These

www.manaraa.com

nodes add capabilities for temperature, acceleration, humidity, barometric pressure, and light monitoring – hardly helpful when one wishes for a simple push button.

4.2.4 Power

A mote's source of power may impact the cost of research should a mote require batteries and may affect the mobility of a mote if it must be tethered to a power source. The Crossbow MICAz is powered by 2 AA batteries that may need periodic changing when used by a device with un-optimized code. The Freescale MC1321x may be powered either by 2 AA batteries, by wall adaptor, or by USB. The Sun SPOT may powered by USB or by an internal battery that must be charged via USB. All the motes are suitable in regards to power for security research.

4.2.5 Housing

The housing of a device affects the developer's ability to interact with the device. A lack of housing increases the probability the device circuitry will be damaged by static; particularly as software developers rarely wear grounding straps. However, the housing itself may prevent a developer from being able to read the user interface, press the device buttons, access the device's break-out pins, or make it difficult (where applicable) to change the device's batteries. The housing must be designed with a careful balance of attention to protecting the device while permitting the user to handle the device.

As-is, the MICAz is difficult to safely handle without fear of causing static damage (Figure 4-1); only the battery casing and sides of the circuit boards may be safely touched without grounding the developer. Crossbow offers an injection-molded housing for the MICAz as an accessory product (Crossbow Technology Inc., 2008a). As

www.manaraa.com

previously discussed the mote essentially lacks buttons and LEDs, so a bubble enclosure is suitable for the MICAz.

Figure 4-1- Crossbow MICAz

The Freescale MC1321x SRB and NCB offer the best, plastic housing of the evaluated kits (Figure 4-2). The clear, top-half of the housing allows easy viewing of the mote's LEDs and allows easy access to the mote's buttons. The housing also has openings for each of the device's power connections, the USB interface, and break-out pins. The only inconvenience of this housing is removing the stiff, battery cover.

Figure 4-2 - Freescale MC13213 SRB

www.manaraa.com

The SPOT housing is pretty, but is wanting in functionality (Figure 4-3). The casing and fin allow for easy handling of the device. The case offers an opening for the USB interface and has a built-out power button. The dark plastic can make it difficult to read the LEDs at a glance – particularly if some messages are transmitted by turning the LED off or if the LED brightness is not at full capacity. Inconveniently, the user must remove the casing cover to access the mote buttons, easily view the LEDs, or work with the break-out pins.

Figure 4-3 - Sun SPOT

4.2.6 Device Types

The available hardware platforms often consist of multiple types of devices. Typically this will include a "base station" type device used for network communication and for programming devices over the air, and sometimes specialized mote offering additional sensor boards or communication interfaces. Each of the evaluated kits offers both regular motes and specialized motes.

www.manaraa.com

A Crossbow MICAz starter kit includes of 3 wireless modules, 2 sensor nodes, and a base station node. The base station is a modified MICAz device that connects to a desktop via USB in order to program network devices and relay messages between the desktop and the network. The regular MICAz motes lack a USB interface or serial port, so they are restricted to communicating with the desktop OTA via the base station.

The Freescale offers a kit that includes two different devices: 2 SRB and 1 NCB. The NCB is designed to act as the network coordinator by adding an LCD and additional break-out pins to the SRB capabilities (described in Table 4-2). The NCB LCD is particularly useful for printing debug messages without having to transmit a message to the desktop and without developing codes for LEDs. Freescale also offers a kit that includes 3 SRB devices and no NCB devices.

The Sun SPOT kit includes two different devices: 2 full SPOT devices and 1 base station. The full SPOT device includes battery, a sensor board, a processing board, and the plastic housing. The base station includes only a processing board and the plastic housing. The purpose of the base station is to primarily act as a network link between the desktop and the network: deploying software updates, gathering responses, and forwarding network data to the desktop. (Notably, motes can speak directly with the desktop if they are connected via USB.) Sun provides software that allows the desktop to emulate a full SPOT device. Inconveniently, the base station is difficult to use without a connected desktop as it has no sensor board (i.e. sensors, LEDs, or interface buttons) and must be powered by USB at all times.

www.manaraa.com

4.3 Development Environment

Like a strong nail and a solid hammer help an individual to create a table from wood, a good development environment with debugging capabilities help an individual to program and to deploy a secure network. A development IDE will provide a clean interface to control and manage multiple files and may provide debugging tools to aid a user in resolving software bugs. A clean deployment system will minimize the difficulty of distributing the software changes to the network nodes.

Crossbow does not provide or recommend a particular development environment. It is left to the developers to choose their favorite C editor (be that an IDE or a text editor) and use Linux (or Cygwin) to build and deploy the code. Fortunately, many tools are available at no cost. Developers with little unaided C experience will struggle as they attempt to determine how to write, build, and deploy code through gcc cross compilers.

The Freescale education mote kit ships with an educational copy of CodeWarrior HC(S)08, yet Freescale's BeeStack is too large be compiled by the restricted, educational software copy (i.e. not even the demonstration software configured with mesh networking and security can be compiled without significant modification). Typically, CodeWarrior with HC(S)08 support costs $950 (Freescale Semiconductor Inc, 2009b). CodeWarrior itself is a common IDE for developing in C, but the real difficulty in departing from CodeWarrior is obtaining a compiler compatible with the MC1321x processing board. The developer may find some frustration in that a CodeWarrior tool is used to build and deploy the code that often fails if multiple instances are running. Further frustrations may be found when the compiler reports a link or compile error using a CodeWarrior error code that provides little to no information regarding the bug.

www.manaraa.com

Sun recommends using the NetBeans IDE and the Apache ANT Java-based build tool to develop code for the SPOT. Any Java editor may be used to develop code, but Sun offers a SPOT-specific module for NetBeans that enables users to build and deploy code to the mote with two clicks in the IDE. ANT may be used for fine control of the code build and deployment. NetBeans offers several debugging tools common to IDE applications, yet NetBeans does not provide an easily read tutorial on how to use the debugging tools. Similarly, users new to ANT may not understand how to configure the build configuration files; however, Sun does provide an excellent default build.xml and provides a good SPOT-specific introduction on how to use ANT. Both of these software tools are available online at no cost.

4.4 Software

The available software for a device may ease the path of development by providing solid ground for a developer to being work. The language of the software may influence the speed developers learn to use the software. The Radio Stack software is critical for programming radio streams and handshaking between two devices. Any available demonstration code may show the developer how a certain library is best used and may provide a template for the developer's custom code. Additional libraries available to the user will enable a developer's ability to program more complex security policies. The Sun SPOTs use a JVM to execute the developer's Java code on the hardware. Finally, an operating system may handle software tasks for the user.

www.manaraa.com

4.4.1 Software Language

The language of the kits was disregarded as a deciding factor in the preliminary selection of these kits, yet language can make a difference to a developer. Just like individuals find it easier to communicate in their native tongue; a programmer may find it easier to program in one language or another. The choice of language is particularly notable when a programmer must choose between C and Java.

C is the language most commonly used in embedded system development. This is true for both the Crossbow and Freescale products. C is best suited for working directly to control hardware. Developers more familiar with procedural coding and C headers will be most comfortable programming these devices.

In recent years it has become common to teach computer students Java as their first language, and many find it difficult to move to the conventions of the C languages; particularly when they are unaccustomed to thinking in procedural or state-machine architectures. The Sun SPOT is a novel device that uses the Squawk JVM to run Java *directly on the hardware*. Java users will be particularly happy with the familiarity of the Java class structure and the ease of using System.out.println() to send a message through the UART.

4.4.2 Radio Stack

The Radio Stack implements the 802.15.4- or ZigBee-compliant source code for communicating between separate nodes. The functions and variables made available by this code will enable a developer to interact directly with the network joining, handshaking, and packet formation of network communication. A radio stack that cannot be easily interfaced with may be detrimental to a developer researching network security.

www.manaraa.com

Crossbow ships with the TinyOS code which is a small, open source Operating System. TinyOS includes a stack that transmits 802.15.4-compliant frames, but does not adhere to 802.15.4 physical and data link definitions (TinyOS, 2009). Any datastream security will be implemented using separate TinyOS libraries.

Freescale offers its ZigBee-2006-compliant BeeStack. A 90-day trial license for this software is included in every Freescale development kit. The BeeStack is a closed-source library that cannot be modified. The BeeStack communications and network configurations may be configured using the BeeKit software available through Freescale requires most of the mote memory. The BeeStack code on the device may require 17-35KB of memory; that is, the code requires 28% to 58% of the memory available to the SRB depending on networking and security settings leaving as little as 29KB for researcher object code and other memory objects. (Freescale Semiconductor Inc., 2007a) Thus the BeeStack (as of 2007) is not appropriate for developers seeking to modify protocols and handshaking or include their own security algorithms.

The Sun SPOT library includes an 802.15.4-compliant communications stack. The stack utilities are available through the Radiogram and Datagram libraries. The lack of a ZigBee-compliant library may hinder the developer as the 802.15.4 does not include any basic security functions for key transport or storage. Fortunately, all the SPOT code is open source and an additional security library is available (see Section 4.4.4), so a determined developer may achieve their objectives.

4.4.3 Demonstration Code

Demonstration code provides a template and a tutorial to the user. Well-written demonstration code provides the user with a good understanding of how code for the

www.manaraa.com

mote is structured. Modular demonstration code can lend tools to the user for reuse such as sensor board commands or radio stream initialization. Demonstration code may also serve as a template for users to begin their own code by editing and revising particular sections of the code.

Crossbow does not provide any demonstration code, so the developers must rely on the device datasheet, their knowledge of C, and code available online from both Crossbow and other developers. Researchers familiar with the software tools may not experience any troubles, but any researcher not firmly grounded in C tools and coding will become frustrated as they struggle to launch an OTA, "Hello World," application.

Freescale does not provide any demonstration code *per se*; the only code provided by Freescale is generated by the BeeKit configuration and a few demonstrations may be inferred from the BeeKit and BeeStack documentation. This code may be overwhelming to the new developer as it consists of several hundred files and several pre-compiled files. The "main" function of this code initializes the BeeStack and application state machine, and the initialization of the main function for the user application is located in a file called "beeapp.c" This compiled code includes a small application that will transmit an incrementing integer over the radio stream to another device to be displayed on that device's LED array. The code is not well commented, but a developer with basic programming experience should be able to understand from this code the basics of interfacing with and LCD and transmitting an integer.

Sun Labs offers the best demonstration application code. The demonstration clearly conveys the potential of the motes with "ectoplasmic bouncing ball" code: when a new node joins the network it communicates with the existing devices to choose a unique

www.manaraa.com

ball color and allows users to "bounce" each ball between any two devices on the network by actuating the accelerometer. Sun's code includes simple libraries for controlling LEDs and the sensor board providing excellent functions to invoke. The template demonstration application used in the tutorial documentation is written clearly and makes it obvious how and where to create custom code.

4.4.4 Additional Libraries

Additional libraries and code may enable a researcher to focus on developing code specific to their topic without having to first "reinvent the wheel." For example, a researcher studying the propagation of key updating is better served by an existing key storage and updating system than writing code from scratch.

Crossbow's additional libraries are created primarily by the online, TinyOS community. This community provides a great deal of source code for the MICAz and other Crossbow motes. The most notable of the additional libraries is TinySec (fully discussed in Section 2.4.1) that adds access control, encryption, and key management to TinyOS. Notably, the University of Cambridge assessed TinySec and found it incompatible with MICAz, but they have since released their own edition that is compatible with the Crossbow device (Cvrcek, 2008). Inexperienced developers may find it difficult to learn to work with TinyOS as it uses NesC instead of regular C and much of the existing online documentation and wikis are incomplete or written by knowledgeable individuals who have forgotten the struggles of a fledgling TinyOS programmer.

There are no known additional libraries or source code specifically for the Freescale MC1321x. The absence of libraries indicates a lack of adoption by the research community and limited public-release research on the part of Freescale. The developer

www.manaraa.com

will be able to find generic C libraries for use on the mote, but (as previously discussed) may encounter difficulties in integrating this code with the BeeStack particularly in the limited resources of the mote.

Sun Labs has released additional, open-source libraries for the Sun SPOT. The libraries do include the aforementioned core, network, and sensor board libraries. What is notable is the ongoing development of a security library for the Sun SPOT. This library is available for free to guest users through an online subversion repository. This library currently includes tools for SSL/TLS handshaking, session establishment, ECC encryption, and HTTPS. The downloadable code includes a read-me file giving a general explanation of the code, yet there is little online documentation or discussion regarding the development of this library. Researchers unfamiliar with SSL, ECC, or the Sun SPOT may find a steep learning curve as there is no supporting documentation.

4.4.5 Java Virtual Machine

The Squawk Java Virtual Machine enables a Sun SPOT to run Java code "on the bare metal" (Simon et al, 2006). JVM itself provides an embedded system developer with garbage collection, pointer safety, thread management, and exception handling. Squawk provides several OS-level mechanisms: the handling of interrupts, network stack functionality, and resource management. The JVM itself takes up very little of a SPOT's resources. The Squawk 1.1 VM and CLDC require a total of 512 KB. The SPOT core libraries require 156 KB. This leaves 3.4M of memory for the research code.

www.manaraa.com

4.4.6 Operating System

An Operating System enables a developer to allow the system to control batch processes, timesharing, process control, memory management, resource allocation, scheduling, and logging. An OS for an 802.15.4 or ZigBee device must consume little of the system's scarce memory and consume few of the processing cycles of the relatively slow CPU. Many developers may elect to include an Operating System on their motes to control multiple applications and provide better memory and resource management.

Many Operating Systems for lightweight embedded systems exist:

- Contiki (Dunkels, Grovall, Voigt, 2004),
- Mantis (Bhatti, et al, 2005),
- SOS (Han, et al, 2005),
- TinyOS (Levis, et al, 2005), and
- ZWOS (Melkonian, Wu, 2005).

Of these, TinyOS is the de facto OS for ZigBee applications. The remaining Operating Systems are rarely implemented outside of the original development group. The TinyOS is quite specific to the Crossbow products, including the MICAz. The greatest challenge of TinyOS may be its use of the NesC language rather than a more common variety of C.

Despite the existence of many possible Operating Systems, an OS is not strictly required to implement a WSN and sometimes is not even available. There is no existing Operating System implementation for the Freescale and Sun motes.

www.manaraa.com

4.5 Analysis of the Security Scenario

The 802.15.4 standard and the ZigBee specification both lack critical security definitions. Neither standard provides any means of authorizing a mote, establishing a key, or updating a key. The current security definitions rely on a key that is preloaded on a mote by either hard coding or by configuration. This security weakness may be exploited if a mote is captured or if unencrypted code is obtained by an unauthorized user. The hypothetical security policy increases its network security by first requiring the user to verify the identity of the mote (i.e. a button press) and then by distributing the master key to the mote. This eliminates the need for a pre-installed key. Further, the hypothetical security policy by providing the utilities necessary to periodically update the master key; this enables the developer to adhere to current best practices.

In the standard implementation, when a device wishes to join a network it simply establishes a link with the nearest router thus joining the network cloud. A more secure system for a home application would require a user to physically authorize a device to join the network, “Yes. I see my toaster wants to join my home network, so I now press this button on the toaster to indicate that I approve.” The networked device is then recorded as a valid mote until the device is removed by the user.

A user must physically press a button on the Coordinator mote as well as the mote wishing to join the network. The strategy reduces the likelihood of an unauthorized mote entering the network as it requires the user to handle each of the motes. Difficulties may arise if the network coordinator and the motes are physically separated; and this may require one user to be present at each of the devices. This strategy assumes that at least the Coordinator is physically protected and not accessible to an unauthorized individual.

www.manaraa.com

The proposed security scenario transmits the master key in a packet after the user has verified that the requesting mote may be permitted into the network. On one hand, this enables the coordinator to provide the latest version of the network key without relying on a default key. One the other hand, since this initial communication occurs by plaintext a packet sniffer will likely be able to capture the key. This key initialization can be strengthened by requiring that the initial communication be established by a wired link, at the cost of user convenience. A mote USB interface or a mote I/O pin may provide the wired link.

An application which periodically updates the master key strengthens its network security. Over time, as greater amounts of encrypted data are passed with a static key, the cipher becomes vulnerable to cryptanalysis techniques that require large data sets. The proposed security policy provides utilities to update the network key to all motes that have formerly been provided with the network key; this will enable an application to initiate the propagation of new key throughout the network without requiring any user involvement.

The hypothetical security scenario will enable the Coordinator to periodically update the network key on all of the authorized motes. The frequency of this update is application-dependant on several parameters: the amount of traffic on the network, the level of security required by the network, and the amount of time required to propagate the new key throughout the network.

A mobile mote's network key will become out of date if is away from the network for a key update. The probability of this occurring depends on the application design

www.manaraa.com

which includes the definitions of the duration and frequency at which a mote may be away from the network and defines the definition of how frequently a new key is created.

To receive the updated key the motes must know the prior network key. This allows the new key to be securely transmitted using the prior network key and only to transmit the key to motes having received prior authorization to be on the network.

The establishment that a mote be loyal to the Coordinator is not fully discussed or realized in the hypothetical security scenario. This loyalty can ensure that a mote is removed from a network when directed by the coordinator (the key utilities for this are provided in ZigBee-2007). Further, it can be used to command obedient motes to scrub their memory of any prior network keys – thus ensuring prior and current keys are not revealed to unauthorized entities.

4.6 Suitability to the Security Scenario

It is crucial that a mote can accommodate the key security components that have been identified as necessary to the developer: packet header access, timers, and ready encryption algorithms (see Section 3.2.6). These components may rely on both hardware and software; for example, a timer is an implementation within hardware, yet the user must have proper software and headers to give register access to the timer.

4.6.1 Access to 802.15.4 Headers

TinyOS and thereby MICAz only offer 802.15.4-compatible frames rather than a full implementation of 802.15.4 (TinyOS, 2009). A small memo exists with documentation of these frames (Hui, Levis, Moss, 2009), but there is no other

www.manaraa.com

documentation describing access or manipulation of these frames. A developer can discover how to manipulate the packet headers by exploring the open-source code.

The Freescale BeeStack does not provide any access to the packet headers. The appropriate headers and functions are inaccessible to developers, due to the closed nature of the code. Thus the MC1321x is impractical for security developers not wishing to discard the existing software framework and begin from scratch.

The Sun SPOT core and security code does not provide any direct method to read or to manipulate the packet headers. Determined developers will likely be able to modify the open-source code to suit their needs.

4.6.2 Timers

The ATmega 128L used by the MICAz has two, single-channel, 8-bit timer/counters. The TinyOS and TinySec documentation are each unclear how many of these timers are reserved and how many are available to the developer. A proposal for a TinyOS revision comments that the OS does not do a sufficient job exposing the microcontroller through interfaces and that the area of timers ought to include several improvements including a need for a counter and a periodic event scheduler (Sharp, Turon, and Gay, 2007). The internal hardware is accessible by the developer when using the C language, but TinyOS itself does not provide any interfaces to the MICAz timers.

The MC1321x has one 5-channel, 16-bit timer/PWM module and one 3-channel, 16-bit timer/PWM (Freescale, datasheet); the BeeKit documentation isn't clear on how many of these timers are claimed by BeeKit itself and how many are available to the developer. The timers are easily used for interrupts, but the developer can expect to run into difficulties measuring time intervals as the timer frequently rolls over. Freescale

www.manaraa.com

provides structures for the handling of rollovers, but relies on the user-written application to manage them.

The Sun SPOT processor has two AT91 Timer/Counters. Each timer counter offers three, 16-bit channels; two of these counters are reserved for system use and the remaining four counters are available to the developer. Sun provides applications notes on how to measure a time interval and how to perform a periodic task using an interrupt (Sun Microsystems Inc, 2007). The difficulty, like the other platforms, is that the timer interval is often much shorter than what the developer may wish to measure; as is the case of the hypothetical security method where a device may wish to wait "10 seconds" for a user to press a button.

4.6.3 Encryption Algorithms

TinyOS as provided by Crossbow does not provide any encryption algorithms; however, the OS may be supplemented with additional code such as TinySec. This module includes support for the SkipJack algorithm, yet documentation describing the use of the code is readily available. It must be noted that TinySec is not necessarily compatible with 802.15.4- and ZigBee-standard radios, so a version ported to the MICAz must be used (Cvrcek, 2008). Determined developers will be able to modify the open-source code to suit their needs.

Freescale offers an 802.15.4-compliant stack for the MC1321x, and it includes an AES algorithm. However, including this code on the device may require 17-35KB of MCU memory depending on the configuration of the security and the network (Freescale, 2007); that is; the code requires 28% to 58% of the memory available to the SRB depending on networking and security settings (Freescale Semiconductor Inc., 2007a).

www.manaraa.com

Unfortunately, it is impossible in the BeeKit to configure which portions of the security utility a developer wishes to include. Furthermore, the interfaces to the built-in utilities are extremely limited, almost to the point of simply being "turn the security on," "ok, now turn it off," and "here's the datagram to encrypt – work your enigmatic security magic." Understandably, Freescale may anticipate selling the BeeStack and BeeKit as a ready-made solution for commercial developers and therefore do not wish the details (and thereby weaknesses) of the security code to be known to the general public; however, the black-box code renders the BeeStack and BeeKit unusable to the individual wishing to study and improve the security code.

The Sun libraries do not ship with any encryption-ready code; however, an open-source, Sun SPOT security library may be downloaded by the developers (Goldman, Meike, Gupta, 2009). This security library for the Sun SPOT includes support for ECC and RSA (Schneier, 1996). (The documentation acknowledges that running RSA will severely slow the system.) A readme.txt documentation file only provides a quick list of instructions on how to build a new SPOT library that includes the crypto code. A short Javadoc API includes a few examples on how to establish a secure radio stream.

4.7 Development Assistance

Documentation is critical in aiding a developer to learn new hardware. Tutorials may guide the user in completing common development tasks such as building and deploying code or transmitting OTA data. Datasheets may provide insight into how the device functions. Software documentation manuals describe the utilities and functions

www.manaraa.com

available to the developers. Finally, demonstration code may clarify how to use functions by illustration and often provide a template to building more complex programs.

4.7.1 Tutorials

Tutorials, Owner's Manuals, and Quick Start guides are essential to aiding a developer in learning a new platform. Crossbow does not provide any form of "getting started" manual to guide a new developer. The TinyOS community does provide a few tutorials and tips on how to begin, but often these make inaccurate assumptions about the new developer's understanding of the hardware, the software, or the NesC language. (NesC is a component-based and event-driven C-derivative language developed for WSN embedded systems. (Gay et al, 2003)) Freescale offers a brief introduction, teaching how to use CodeWarrior, and Freescale BeeKit provides a tutorial on how to launch a basic application, but does not provide much in way of further assistance. Sun provides a Quick Start guide that walks the new user through the development process: from building and deploying (both in the IDE and via ANT) to the basics of using the base station. Further manuals are available that walk a new developer through each of the sensor board components and through using the radiostream.

4.7.2 Data Sheets

Datasheets provide a summary of the characteristics of the device hardware. This is particularly useful when working in C where a developer may often need to reference how to access a certain pin or sensor. The MICAz datasheet provides only a summary of the basic hardware features and no details about the hardware design itself. The lack of published information may frustrate developers needing more detailed information about

www.manaraa.com

the hardware, yet these details might be found in publications written by the online community. Freescale's MC1321x datasheet is excellent documentation of the hardware features and thoroughly explains the details of how the timer and ADC function. Developers may find the Freescale datasheet to contain helpful hints on how variables or functions may be named in the provided libraries. The Sun SPOT, being written in Java, lacks the ability to directly address a portion of the microcontroller in the same sense that C does, but the Sun libraries do provide excellent software interfaces. No true datasheet exists for the Sun SPOT, but the documentation manuals do include example code on how to set IRQ Timers and configure LEDs.

4.7.3 Software Documentation

The functions and classes unfamiliar to a developer should be described by software documentation such as manuals or APIs. Crossbow's documentation appears to primarily reside in the community-written, Tiny-OS wiki. Freescale documents their software in a long series of PDF documents that are so extensive that a PDF document is required to explain which document manuals are which. The Freescale manuals do appear to document every function within the BeeStack, but the sheer amount of documentation in pdf format makes it cumbersome to locate the desired function – even with an OS search function. The Freescale documentation also lacks clear examples on how to use a function and often neglects recording that function *x* ought to be called to configure library *q* before attempting to invoke function *y*. The Sun SPOT software API documentation is easy to navigate for those familiar with the typical javadoc output – all functions are categorized within their libraries and jars and an extensive index of hyperlinks and a search engine are provided for those in doubt. The developer may find

www.manaraa.com

difficulties with the Sun SPOT documentation as it isn't as easily browsed as a PDF or a wiki.

The difficulty with the Sun SPOT documentation is that if you don't know what you're looking for, it is difficult to virtually thumb through various topics like it would be possible to do in a PDF or a wiki.

4.7.4 Online Community

An online community actively discussing system development may help smooth the path for an inexperienced developer. The presence of an online researching community indicates the acceptance level of hardware platform for research. Such communities may be manifest in online tutorials (often published in blog format), in discussion forums, or through community-editable wikis. (Table 4-4 provides a summary of the notable online communities.)

The Crossbow MICAz community is concentrated within the TinyOS community. The hub of TinyOS is tinyos.net, where a documentation wiki, TinyOS tutorials, library work groups, and a documentation wiki may all be found. A "tinyos-help" mailing list links developers together to help answer questions and find resources; this mailing list is archived so historic question and answers may be referenced before repeating a question. Researchers at various universities also frequently publish on topics about the MICAz, but it seems it is rather uncommon to publish one's code online.

The Freescale WSN community is very small and has not grown in recent years. The only community for the MC1321x is the Freescale forums, and those consist of more individuals requesting help than individuals offering answers. The sense within the community is that the MC1321x is more meant for creating applications than it is meant

www.manaraa.com

for the researcher. This sense may be derived from the black-box stack, as discussed earlier.

The Sun SPOT community is primarily an applications and hobbyist community. A search on YouTube reveals many developers eager to show off their innovations. The Sun SPOT world forum acts as a community center, and frequently answers the questions of new developers. Sun Labs directs most of the security research using Sun SPOTs, and periodically releases new updates to their security library.

Table 4-4 - Online Communities

Company	Community	URL
Crossbow	TinyOS Community	http://tinyos.net/
Freescale	Freescale Forums: 8-bit MCU	http://forums.freescale.com/freescale/board?board.id=8BITCOMM
Sun	Sun SPOT World	http://www.sunspotworld.com/forums
Sun	Sun SPOTs Projects	https://spots.dev.java.net/
Sun	Sun SPOT Libraries	https://spots-libraries.dev.java.net/

4.8 Other Tools

Additional tools may be available to the developer. Simulation tools may be particularly useful to developers seeking to test the scalability of a system. Emulating tools may likewise be useful for testing a prototype implementation and to test code before deploying it to a mote. Background debugging interface devices may be useful for debugging code on the mote itself. Bench marking tools can measure a security policy's impact on mote resources.

www.manaraa.com

4.8.1 Simulators

Simulation of a network may provide data models regarding the feasibility of the network. Simulators are a low cost, more practical for testing, and easier to implement than a deployed network. Yet the limits of a simulator must be accounted for when evaluating data provided by a simulator. Researchers often question the accuracy of simulations stating that such programs' models of wireless propagation do not accurately reflect real-world behavior (De, 2007; Heidemann et al, 2001) and that such programs do not sufficiently create accurate values for lower layer phenomenon (i.e computation time, packet loss, and routing) (De, 2007).

Dozens of wireless simulators are available and used within WSN research (Becker, 2007), yet only five are compatible with 802.15.4: TOSSIM, ATEMU, Avrora, DiSenS, and NS-2. Of these only a few are compatible with the MICAz, and none are compatible to the MC1321x or Sun SPOT. TOSSIM (Levis, et al, 2003) is tailored specifically to MICA motes running TinyOS. ATEMU (Polley, et al., 2004) and Avrora are specific to the MICA2 platform; likewise, Avrora (UCLA Compilers Group, 2007) and DiSenS (Wen, 2005) are designed for the MICA2 and MICAz motes running TinyOS.

Only NS-2 (Marandin, 2007), a popular simulator for 802.15.4 applications, seems able to simulate ZigBee networks without being dependent on TinyOS or a specific hardware platform. NS2 has been used to simulate a mixed AODV and Tree Routing algorithm within a ZigBee network (Ran, Mao-heng, Youmin, 2006), and to simulate and measure the throughput of a ZigBee network (Burchfield, Venkatesan,

www.manaraa.com

Weiner, 2007). It may be possible to simulate the Freescale or Sun devices with NS-2, but there is no available documentation verifying such.

4.8.2 Emulators

Emulators enable one type of hardware (i.e. a PC) to function as another type of hardware (i.e. a mote). The use of an emulator is twofold. First, an emulator may test a developer's software and identify bugs before the time is taken to deploy the software to the networked motes. Second, an emulator may enable to user to test the scalability of a solution by allowing readily available hardware to masquerade as a specialized mote. Of the examined kits, only Sun offers a mote emulator. The software, called the Sun SPOT Emulator, provides a GUI interface to enable a base station connected to the PC to act as a full device.

4.8.3 Background Debug Interfaces

A Background Debug Module (BDM) is an embedded system that assists developers in debugging mote software while the code is running on the mote. In-circuit debugging enables users to test their programs under deployed conditions and under the mote restricted hardware. Only Freescale offers a BDM device for debugging its motes – and this device doubles as the mote program loader.

4.8.4 Benchmarking Tools

The actual measurement of a WSN network is challenging: environment variables are difficult to repeat, motes often cannot spare their limited memory for benchmarking software, and any data that is collected must be transmitted to a host computer (Park,

www.manaraa.com

Chou, 2006). For example, the analysis software LMBench (McVoy, 1998) requires a minimum of 16 MB of memory and 16 MB of program space to monitor the mote state and resources.

To date, only two 802.15.4 benchmarking suites exist: EmPro and Daintree Network's Sensor Network Analyzer. Both of which are ZigBee-compliant as well as 802.15.4-compliant. Either device may be used with the Crossbow MICAz or the Freescale MC1321x. (The Sun SPOT is not ZigBee-compliant and therefore is excluded from these benchmarking tools.)

EmPro (Park, Chou, 2006) claims to "monitor every activity of each wireless sensor node." The authors strive to create an emulation system that controls the environment inputs in order to provide a repeatable means to compare WSN hardware platforms. EmPro's energy consumption and packet error rate measurements would prove useful to an evaluation of the hypothetical security solution. The system is created for WSN hardware and can likely be used regardless of what runs on the networking and application layer. Unfortunately, EmPro requires additional hardware required for emulation, and the system EmPro appears to have been abandoned since Chulsung Park achieved his PhD.

Daintree Network's Sensor Network Analyzer (SNA) (Daintree Networks, 2007) is widely used within industry to debug, deploy, and manage ZigBee networks. SNA uses specialized sniffer hardware to analyze and collect data in the ZigBee Network; therefore it is compatible with all ZigBee devices (including those which use the ZigBee-2007 specification). The SNA Pro edition collects network packets to show packet contents, a packet time line; analyzes the NWK and APS layers; reports on the ZigBee-2004 and

www.manaraa.com

ZigBee-2006 security; creates extensive graphic representations of the network; provides "comprehensive numerical statics" and visual statistics; maintains the network by automatically detecting nodes; and provides management tools to control the clusters, binding, and mote participation within the network. Unfortunately, the cost of the standard addition software and the Sensor Network Adapter hardware is approximately $1,000.

4.9 Expense

The cost of a platform is an important consideration when budgeting and proposing research. Generally, the Crossbow MICAz, Freescale MC1321x, and Sun SPOT devices are in similarly priced, but the total cost of research may be impacted by the security policy experiment design and research tools. The total cost may be impacted by the number of kits a researcher must purchase to acquired the desired number of motes and base stations. The total cost may also be impacted by the inclusion or exclusion of necessary software.

The Crossbow kits may include several Crossbow products. The corresponding data sheets refer to MICAz "sensor nodes" which each consist of a processing board and a sensing daughterboard and there are MICAz "base stations" which each consist of a processing board and PC interface daughterboard. The kits may also include a data acquisition daughterboard which has 11 channels of 12-bit ADC input, a temperature sensor, and a humidity sensor.

Crossbow offers three development kits for the security researcher to consider (see Table 4-5)(Crossbow Technology Inc, 2009a,b,c). The Starter Kit, the smallest

www.manaraa.com

available, includes 2 sensor notes, 1 base station, and a copy of the MoteView software. The Professional Kit includes an additional 5 sensor nods and a data acquisition board to the Starter Kit. The Classroom kit includes 20 sensor nodes, 10 base station nodes, 10 seats of MoteWorks, and additional classroom materials; this kit is only available by contacting the sales department. Additional MICAz sensor nodes may also be purchased for $275 each. The Crossbow is a cost effective platform for researchers to consider – especially as the development and compiler incur no additional cost.

Table 4-5: Crossbow MICAz Development Kits (Crossbow Technology Inc., 2009a,b,c)

	Starter Kit	Professional Kit	Classroom Kit
Sensor Nodes	2	6	20
Base Station	1	1	10
ADC Board	0	1	0
Software	MoteView	MoteView	10 Seats of MoteWorks
USD	$795	$2,195	unpublished

Freescale offers seven kit selections – each differing slightly from one another (see Table 4-6)(Freescale Semiconductor Inc, 2009a). The most basic kit is the Developer Start Kit that includes 2 SRB devices, a limited version of CodeWarrior, and a 90 day BeeKit license. This kit is best as a supplement kit, as it does not include any NCB motes and does not include a BDM (used to program and debug the motes). On the other end of the scale, the most deluxe kit is the ZigBee Development Kit that includes 4 SRB motes, 3 NCB motes, 1 BDM, a standard edition of CodeWarrior, a full BeeKit license, and a standard edition of Daintree.

www.manaraa.com

Table 4-6: Freescale MC1321x Development Kits (Freescale Semiconductor Inc, 2009a)

	Develop. Starter Kit	Develop. Starter Kit -BDM	Consumer Starter Kit	Nwk Starter Kit	Nwk Starter Kit -BDM	ZigBee Eval. Kit	ZigBee Eval. Kit -SFTW
SRB	2	2	1	2	2	4	4
NCB	0	0	1	1	1	3	3
BDM		X	X		X	X	X
IDE	α	α	α	α	α	α	β
BeeKit	ε	ε	ε	ε	ε	ε	θ
Daintree						Basic	Stnd.
USD	249	349	399	499	549	1,749	3,299

α - CodeWarrior Special Edition
β - CodeWarrior Standard Edition
ε – 90 Day BeeKit License
θ – Full BeeKit License

It is very likely a researcher will find a kit or a combination of kits that will provide the desired number of motes and base stations. The cost of the Freescale motes lies within the range of the other motes. The great difficulty is that the limited CodeWarrior edition is *too* limited and cannot compile the amount of code necessary to include security algorithms. CodeWarrior with HC(S)08 support costs $950 (Freescale Semiconductor Inc, 2009b). Further, the BeeKit license is only valid for 3 months. Academic researchers without additional uses for the IDE or the compiler may find that the Freescale kits may be both too costly and academic researchers may find the BeeKit license restricts them to too short a development and testing window.

There is only one development kit selection offered by Sun. The SPOT Development Kit is comprised of 2 full SPOT devices, 1 base station device, and a software CD (see Table 4-7)(Sun Microsystems Inc., 2009a). The CD contents include development tools, demonstration code, and Squawk JVM. Developers seeking to create

www.manaraa.com

larger networks will be find the offerings of this kit bothersome as the base station does not have any sensors and does not have a battery, so it can only serve as tethered mote in narrow applications.

All of the development software and programming code necessary are available at no cost. The published educational discount cost of the kit is $299 (Sun Microsystems Inc., 2009b). This is a very cost-effective kit for academic researchers to consider.

Table 4-7: Sun Spot Development Kit (Sun Microsystems Inc., 2009a)

	Development Kit
Full Device	2
Base Station	1
Squawk	X
Software	NetBeans
USD	$750

4.10 Summary

This chapter has provided a detailed comparison of the Crossbow MICAz, Freescale MC1321x, and the Sun SPOT. Various aspects of the hardware, software, security resources, development tools, additional tools, and kit expense were discussed. Chapter 5 shall draw conclusions based on this data.

www.manaraa.com

www.manaraa.com

5 Conclusions

Wireless Sensor Networks are anticipated to become common in future home environment applications. A WSN is a mesh network of very small nodes gathering environment data, sharing this data, and actuating devices according to their programming. Such a network may aid users by monitoring their health, managing their utilities, and assisting with their special needs.

IEEE 802.15.4 is a standard defining the physical and data layers of wireless sensor network communication. This standard is designed to accommodate devices with very small microprocessors running on small batteries with an anticipated lifetime measured in years. The ZigBee specification is implemented atop 802.15.4 to add standardization for the network layer and application framework.

The security of these standards is of particular interest because it is important to protect the information of individuals using a wireless sensor network. The ubiquitous devices may gather information about the users' presence, habits, and identity that should not be accessible to unauthorized entities.

Neither 802.15.4 nor ZigBee are completely secure. Each standard directly states that its definitions on key initialization, key updating, and device group management are either entirely out of scope or only partially implemented. Further research must be

www.manaraa.com

completed to ensure the tools are available to secure these aspects of a wireless sensor network.

Researching and developing for a WSN is not without its challenges. Ideally, the mote processor and memory are very small in order to minimize their impact on a mote's battery lifetime; subsequently the resources available to a researcher to store and run security algorithms on the device are limited. Debugging by tracing communications between devices, identifying faulty motes, and troubleshooting software on its native processor becomes difficult because the code is run on a wireless, distributed, and embedded system.

A researcher may find additional challenges when learning a new hardware platform. Documentation describing how to use a system may be lacking. The hardware may not provide sufficient inputs and outputs to aid the researcher in troubleshooting and designing applications. There may be no existing security code to build security solutions upon.

This thesis has assessed three prevalent hardware kits: Crossbow MICAz, Freescale MC1321x, and the Sun SPOT. Each kit was evaluated for its suitability to academic research. Evaluation parameters included hardware, software, security resources, development environment, documentation, additional tools, and expense.

This final chapter draws conclusions from the data gathered and comparisons made in Chapter 4. First, a summary of all the kit data is presented and recommendations are made to researchers. Second, suggestions are made regarding how to improve the usability of each kit for security research. Third, future research is recommended.

www.manaraa.com

5.1 Summary of Hardware Kits

This section summarizes the data presented in Chapter 4. Table 5-1 provides a quick reference of basic development features of each hardware kit and Table 5-2 provides a summary of the evaluated kits' comparative features. The topics discussed include the hardware specifications, power options, user interface, sensor availability, development environment, software, security libraries, demonstration code, and expense of each candidate platform. Recommendations are made to researchers regarding what kits they ought to consider given certain goals, background, or constraints.

In terms of memory and processing, the Sun SPOT outclasses the MICAz and Freescale MC1321x. The Sun SPOT has at least sixty times as much Flash memory, an order of magnitude more RAM, a clock that's at least 4 times faster, and a 32-bit processor rather than an 8-bit processor. To an extent, this plethora of processing power defeats a primary purpose of a mote – to run on relatively small batteries for a very long time; though no WSN studies have been completed studying the impact of the microprocessor on the battery lifetime. A researcher wishing to design security protocols will prefer the Sun SPOT because the processor and memory will not restrict the size of the implemented code as much as the other two kits will. A researcher wishing to work within a more traditional WSN should consider the MICAz or Freescale MC1321x as these kits adhere to the prevalent notion that WSN devices ought to be small and limited.

www.manaraa.com

Table 5-1: Development Kits Summary

		MICAz	MC13213	SPOT
Hardware	**MCU**	MPR2400	HCS08	ARM920T
	Clock	16 Mhz	40 MHz	180 MHz
	Bits	8-bit	8-bit	32-bit
	Flash	128 KB	60 KB	4,000 KB
	RAM	4 KB	4 KB	512 KB
	Timers	2 8-bit channels	8 16-bit channels	6 16-bit channels
Comm.	**802.15.4**	X	X	X
	ZigBee	X	X	
	Serial	X		
	USB		X	
Power	**AA Batteries**	2	2	0
	Internal			X
	USB		X	X
	Power Supply		X	X
Interface	**LEDs**	2	4	8
	Switches	0	4	2
	Integrated Sensors		X	X
	I/O Pins	X	X	X
	Daughterboards	X		
Development	**Language**	C, NesC	C	Java
	Recommended IDE	none	CodeWarrior	NetBeans
	Build	GCC	HC(S)08	ANT
	Load	Serial or OTA	BDM (USB)	USB or OTA
	Serial	X		
	USB		X	X
	OTA	X		X
S/W	**Stack**	TinyOS	BeeStack	Spot Core Library
	OS	TinyOS		Squwak JVM
	Encryption	Skipjack	AES	ECC

www.manaraa.com

Table 5-2 – Evaluated Kits Comparison

		MICAz	MC1321x	SPOT
Hardware	Largest Processor			X
	Most Flash			X
	Most RAM			X
	Most Timers			X
	AA Batteries	X	X	
Interface	Most LEDs			X
	Most Switches		X	
	LCD		X	
	Daughterboards	X		
Development	IDE At No Cost	X		X
	Compiler At No Cost	X		
	Code Writen in C	X	X	
	Code Written in Java			X
	Security Libaries			
	Open Source	X		X
Documentation	Most Demo Code			X
	Best Datasheet		X	
	Best API			X
	Active Online Community	X		X
	Peer-Reviewed Publications	X		

Timers are expected to be used by the researchers to prevent system deadlock, measure intervals of time, and benchmark their code. The Sun SPOT appears to offer more timer channels than its competitor motes (due to its larger processor), yet several of these timers may be undocumented as being reserved by the Squawk JVM. Researchers must determine how many timers are needed for their research before selecting a platform. One researcher may find the two 8-bit timer channels of the MICAz sufficient, yet another researcher may not notice the number of MC1321x timers exceeding his or

www.manaraa.com

her development code needs, while yet another researcher may find the lack of additional channels frustrating.

The importance of a mote's power source depends upon the application of the security research. Researchers must design or anticipate the power needs of their application assumptions before selecting a mote environment. If an application does not need any mobile or distributed motes than any of the evaluated kits are suitable for the research. If an application does require that a mote has freedom of movement than the researcher must make the power options of a mote a priority. The Freescale MC1321x and MICAz should be used for applications requiring motes with a longer battery lifetime. Researchers wishing to quickly change AA batteries ought to select the Freescale or Crossbow motes; these same researchers ought to avoid the Sun mote as the SPOT's internal battery can only be charged by USB and is not swappable.

A mote's user interface impacts the user's ability to design applications and troubleshoot mote software. The Crossbow MICAz's lacking interface may hinder the developer. This device expects the developer to rely on communications with the mote via terminal rather than plentiful providing onboard inputs and outputs. The Freescale MC1321x NCB is particularly useful for debugging software and communicating with the application user because it offers an LCD. Each Freescale SRB provides 4 buttons and 4 LEDs. The MC1321x is the moderate choice of the evaluated platforms being that its interface is neither limited nor plentiful. The Sun SPOT offers the most glitz with its generous 8, tri-colored LED interface which may aid users in developing output codes. Oddly, the Sun SPOT provides only 2 switches. Experienced developers with a preference for utilitarian motes should consider the MICAz; however, developers with

www.manaraa.com

less embedded system and wireless programming experience should not adopt the MICAz. Researchers preferring an LCD interface should use the Freescale NCB whereas researchers preferring more precise LED communication should use the Sun SPOT. Conversely, researchers requiring more elaborate input options should reject the Sun SPOT in favor of the Freescale.

Researchers should fully understand what types of data they wish to collect before committing to a platform or allow time and resources for designing other sensors and actuators. The MICAz, MC1321x, and SPOT each offer accelerometers, temperature sensors, and ADC either onboard or through a daughterboard. The MICAz offers a wide variety of expansion daughterboards for sale that will likely have a particular sensor a researcher requires (be it anything from GPS to a barometric pressure sensor). Researchers needing specific data collection should avoid the Freescale and Sun motes as neither the Freescale nor the Sun motes offer any expansion daughterboards. Researchers not needing specific data collection may use any of the development kits.

Each platform development environment prescribes different software and offers different methods of code deployment. The MICAz relies on a developers' choice of IDE (if any), a gcc cross-compiler, and loads code via a serial port. The Freescale MC1321x relies on a compiler offered only through CodeWarrior and loads code to motes through a BDM. The special edition of CodeWarrior shipped with most Freescale kits limits the size of the code that a developer may compile; observation indicates that in many cases the code required for ZigBee-compliance and security exceeds that limitation. The Sun SPOT recommends the NetBeans IDE, deploys code over USB through ANT, and uses the Java SDK to build the code for its device. The MICAz and SPOT development

www.manaraa.com

environments are available at no cost whereas a standard edition of CodeWarrior HC(S)08 may need to be purchased for the Freescale kit. In this case, the selection of a kit depends on which environments the researchers are accustomed to, which are already available to the researcher, and what the budget of the researcher permits. Researchers experienced in embedded system development should select the MICAz as it allows them to use their familiar IDE and gcc tools at no cost. Researchers who do not yet own a copy of CodeWarrior may prefer to avoid the Freescale motes due to the high cost of the standard edition of the IDE. The Sun SPOT may be considered by both experienced and inexperienced developers as the SPOT's recommended suite is available at no cost.

Each evaluated platform provides software that is distinct. The MICAz relies on the TinyOS, which is not fully 802.15.4-compliant, to provide the communication stack and task management. TinySec is available to add additional security tools to the TinyOS. The Freescale MC1321x relies on the Freescale BeeStack as configured by the Freescale BeeKit. The BeeStack does include AES encryption, but its black-box approach to code is much better suited to application developers rather than security researchers. The Sun SPOT relies on the Squawk JVM to manage application tasks and run the Java on the mote hardware. The basic SPOT core libraries may be extended using the SPOT security libraries that include ECC encryption.

Each platform readily offers encryption algorithms. The Crossbow, through TinySec, offers the Skipjack algorithm. The Freescale offers AES encryption. Sun offers RSA and ECC. Notably, efficient implementations of ECC are still patented, and there is some question if Sun has obtained the rights to distribute the ECC intellectual property.

www.manaraa.com

Sun also support SSL/TLS which uses public key cryptography to authenticate the network devices.

It is important that a researcher be able to extend the existing security libraries of a platform. TinySec for MICAz is open source, but also very poorly documented. A determined researcher should be able to use this platform. The Freescale MC1321x cannot be easily extended – and therefore should be avoided by researchers who wish to create their own protocols or include their own security algorithms. The SPOT security library is open source, but, like the MICAz, is poorly documented for new-comers to the code. Determined researchers should be able to modify and extend the code. The Freescale motes should be rejected by any researcher wishing to develop more complex security policies or measures. The MICAz should be selected by those researchers who are more familiar with C. Finally, researchers who are more familiar with Java should consider the Sun SPOT.

The demonstration code offered by a platform may help developers understand how to structure and design code for the given platform. The MICAz is lacking in demonstration code, but an excessive amount of code (both good and bad) is available through the active research community. The BeeKit compiles the BeeStack with a basic demonstration application. That demonstration application provides a good template for the overall code structure, but is insufficient in guiding a less experienced developer. Finally, the Sun SPOT appears to be targeted to java developers with little embedded system experience as it provides extensive demonstration tutorials and well-written documentation manuals for that demonstration code. In this instance, those researchers quite familiar with embedded system development may prefer the Crossbow MICAz

www.manaraa.com

whereas researchers less familiar with embedded system development ought to prefer the Sun SPOT.

Finally, the expense of each hardware kit is comparable to one another. The distinguishing cost to the researcher is introduced first by the quantities of motes the researcher wishes to work with and second by the cost of any additional tools or licenses. Freescale may quickly become expensive if the researcher needs to purchase a copy of the HC(S)08 compiler or if the researcher needs to extend the 90-day trial license of BeeKit. The development kits and software for the MICAz and Sun SPOT are available at no cost. Academic researchers should contact each company's sales team for educational discounts before purchasing a development kit.

Overall, a researcher's hardware selection should be based on the needs of the researcher regarding the experiment application and the depth of security development. Crossbow MICAz is best suited to complex experiment applications (i.e. supplementing the processor board with a daughterboard) and can be used for security research; this system is particularly recommended to researchers familiar with C and embedded system development. Freescale MC1321x should be useful to researchers already own the HC(S)08 compiler who wish to only work on application development. Sun SPOT is best suited to simple experiment applications (e.g. bouncing ball) and can be used for security research; this system is particularly recommended to researchers familiar with Java and who have a basic understanding of embedded system programming.

www.manaraa.com

5.2 Recommended Improvements

Each of the examined platforms was found suitable for WSN research, though each researcher may find one of the platforms to be better suited to a particular aspect of security research over the others. Within this section several suggestions are offered on how to improve each of the platforms.

The MICAz needs to improve its accessibility to new users. An additional daughterboard needs to be created with basic user debugging devices: an LCD, multiple switches, multiple LEDs, a buzzer, etc. The documentation of the MICAz needs to become more centralized – the TinyOS wiki is a good start, but lacks details about integration with the MICAz hardware. The usefulness of the existing documentation would improve if they provided more hand-holding and technical explanation to the inexperienced developer; individuals unfamiliar with the conventions of embedded system programming often lack the intuition necessary to understand the documentation. Finally, Crossbow needs to make more demonstration code readily available to the new user.

Freescale needs to make many changes to improve the MC1321x for security research. The IDE and compiler may too costly for some researchers to install – in many cases more expensive than the development hardware itself. If the BeeStack code cannot be opened, more interfaces need to be built into the code to allow developers to write their own handshaking protocols and wrap in their own encryption algorithms. Further, a more searchable code documentation set (such as an API or a wiki) would clarify the existing utilities to new users. A final concern is that the development kits ship with only a 90-day license of the BeeKit, which may not provide enough time for some individuals

www.manaraa.com

to complete their research. Freescale should also find creative ways of fostering a research community through incentives, competitions, introduction of small projects to sponsored universities, etc.

The release delays and care in developing the Sun SPOT are evident in the quality of the kit, yet Sun still needs to seek to improve the platform further. Foremost Sun needs to promote the SPOT in more peer-reviewed research (most projects are currently published through Sun Lab blogs or by hobbyists on YouTube). The Sun SPOT security library also needs more tutorials and documentation describing to the new user how to best implement the files and how to write their own protocols. The ease of using the Sun SPOT security would be further improved by creating a utility that automatically integrates the SPOT security library with the SPOT core library.

In general, none of these development systems have seriously pursued implementing best practices in security within their development kits. Crossbow relies on the TinyOS community for the development of its security features. Freescale provides a measure of security with its BeeStack, but does not provide a means for researchers to evaluate and build-upon the given security. Sun SPOT does have several individuals – mostly Sun Labs members – working on the Sun SPOT security libraries, yet these libraries are not yet developed and documented to a point where new researchers may easily contribute. All the development platforms could use improvement in regards to promoting security research on their hardware.

www.manaraa.com

5.3 Future Work

Many topics worth further investigation were outside the scope of this thesis regarding the state and suitability of 802.15.4 hardware for security research. First, a number of notable kits were excluded for brevity or cost that ought to be examined as well. Second, little research exists cross-examining the performance of the hardware running similar software; extending and implementing the hypothetical security scenario would be an excellent place to begin. Third, the suitability of 32-bit processors (i.e. the SPOT) needs to be tested and measured for viability in an 802.15.4 deployed network. If a 32-bit processor can be used within the power constraints of WSN devices than processing-intensive security techniques formerly rejected by WSN researchers should be investigated.

www.manaraa.com

www.manaraa.com

6 References

Andersson, A., and M. Thoren. 2005. ZigBee, A suitable base for embedded wireless development? M.S. thesis., Chalmers University of Technology.

Bateman, C., J. Armstrong, and R. Helps. 2007. Introducing ZigBee theory and practice into information and computer technology disciplines. In 2007 ASEE annual conference & exposition.

Becker, M. Simulation tool comparison matrix. 2007-07-25 2007. Internet on-line. Available from <http://www.ist-cruise.eu/cruise/Public%20documents/wp123-wsn-simulation-tool-knowledgebase/simulation-tool-comparison-matrix>. [10/25, 2007].

Bhatti, S., J. Carlson, H. Dai, J. Deng, J. Rose, S. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. 2005. MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms Mobile Networks and Applications 10, no. 4: 563-579.

Burchfield, T. R., S. Venkatesan, and D. Weiner. 2007. Maximizing throughput in ZigBee wireless networks through analysis, simulations and implementations. Dallas, Texas.

Cantrell, T. 2008. Got Chips? Circuit Cellar , no. 211 (Februrary 2008) : 78-86.

Casas, R., O. Casas. 2005. Battery sensing for energy-aware system design. Computer 38, no. 11 (Nov 2005) : 48-54.

Cherry, S. 2006. Wi-Fi Nodes to Talk Amongst Themselves. IEEE Spectrum 43, no. 7: 55.

Crossbow Technology Inc. Housings. 2008. Internet on-line. Available from <http://www.xbow.com/Products/productdetails.aspx?sid=162>. [03/30, 2009].

Crossbow Technology Inc. TelosB. 2008. Internet on-line. Available from <http://www.xbow.com/Products/productdetails.aspx?sid=252>. [03/29, 2009].

www.manaraa.com

Crossbow Technology Inc. WSN classroom kit. 2008. Internet on-line. Available from <http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/WSN_Classroom_Kit.pdf>. [03/30, 2009].

Crossbow Technology Inc. WSN starter kit datasheet. 2008. Internet on-line. Available from <http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/WSN_START_Datasheet.pdf>. [03/30, 2009].

Crossbow Technology Inc. WSN professional series datasheet.

CSR. CSR gives world's first public demonstration of ULP bluetooth. 14 April 2008 2008. Internet on-line. Available from <http://www.csr.com/pr/pr330.htm>. [05/13, 2008].

Cvrcek, D.. Security of TinySec. 2008-06-01 2008. Internet on-line. Available from <http://www.cl.cam.ac.uk/research/security/sensornets/tinysec/>. [03/29, 2009].

Daintree Networks. Daintree networks sensor network analyzer (SNA). 2007. Internet on-line. Available from <http://daintree.net/products/sna.php>. [10/25, 2007].

De, P. 2007. MiNT: A reconfigurable mobile muli-hop[sic] wireless network testbed. Ph.D. diss., Stony Brook University.

Digi-Key Corporation. Digi-Key Part Search. 2009. Internet on-line. Available from < http://www.digikey.com/scripts/dksearch/dksus.dll?Cat=3539644&keywords=InSight&vendor=636>. [04/15, 2009].

Dunkels, A., B. Gronvall, and T. Voigt. 2004. Contiki - a lightweight and flexible operating system for tiny networked sensors. In Proceedings on the 29th IEEE international conference on local computer networks (LCN'04)455-462.

Ember Corporation. EM250 single-chip ZigBee/802.15.4 solution. July 5, 2006 2006. Internet on-line. Available from <http://www.ember.com/pdf/EM250_Datasheet.pdf>. [01/14, 2008].

Ferrari, G., P. Medagliani, D. Piazza, and M. Martalo. 2007. Wireless Sensor Networks: Performance in Indoor Scenarios. EURASIP Journal on Wireless Communications and Networking 2007.

Forster, E. M. 2001. The machine stops. In Selected stories. Edited by Anonymous New York: Penguin Books.

Freescale Semiconductor Inc. 1321x development kits product summary. 2009. Internet on-line. Available from <http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=1321x_Dev_Kits>. [03/30, 2009].

www.manaraa.com

Freescale Semiconductor Inc. Development suites. 2009. Internet on-line. Available from <http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=0127262E703BC5>. [03/29, 2009].

Freescale Semiconductor Inc. Freescale's ZigBee technology products summary. 2007. Internet on-line. Available from <http://www.farnell.com/datasheets/22865.pdf>. [03/30, 2009].

Freescale Semiconductor Inc. MC13211/212/213/214 (MC1321x rev 1.1). 03/2007 2007. Internet on-line. Available from <http://www.freescale.com/files/rf_if/doc/data_sheet/MC1321x.pdf>. [05/07, 2007].

Freescale Semiconductor Inc. MC1322x Development Kits. 2008 2008. Internet on-line. Available from <http://www.freescale.com/files/wireless_comm/doc/brochure/BRIEEE802154DKIT.pdf?fpsp=1 > [04/14, 2009].

Freescale Semiconductor Inc. MC13224V (MC1322x rev 1.2). 02/2009 2009. Internet on-line. Available from <http://www.freescale.com/files/rf_if/doc/data_sheet/MC1322x.pdf?pspll=1> [04/14, 2009].

Fusting, C. 2008. Jungles, petabytes, and discoveries: Yggdrasil - A data collection framework.

Gamage, C., K. Bicakci, B. Crispo, and A. S. Tanenbaum. 2006. Security for the mythical air-dropped sensor network. In 11th IEEE symposium on computers and communications, 2006. ISCC '06.41-47.

Gay, D., P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. 2004. The nesC language: A holistic approach to networked embedded systems. In Conference on programming language design and implementation archive proceedings of the ACM SIGPLAN 2003 conference on programming language design and implementation1-11.

Geer, D. 2006. UWB Standardization Effort Ends in Controversy. Computer 39, no. 7 (July 2006) : 13-19.

Girao, J., D. Westhoff, A. Poschmann, A. Weimerskirch, A. Grilo, R. Silva, M. Nunes, A. Casaca, E. Osipov, J. Riihijarvi, P. Steffen, P. Langendorfer, K. Piotrowski, L. Buttyan, G. Acs, P. Schaffer, C. Catelluccia, and A. Francillon. 2006. Scenario definition and initial threat analysis.

Goldman, R., R. Meike and V. Gupta. Spots-security. March 2009 2009. Internet on-line. Available from <https://spots-security.dev.java.net/>. [03/29, 2009].

www.manaraa.com

Google. Netbeams. Mar 26, 2009 2009. Internet on-line. Available from <http://code.google.com/p/netbeams/>. [03/29, 2009].

Goth, G. 2007. Wireless USB: Just the First UWB Battle? IEEE Pervasive Computing 6, no. 1: 8-9.

Gumstix Inc. Gumstix motherboards. 2008. Internet on-line. Available from <http://www.gumstix.com/platforms.html>. [03/29, 2009].

Gupta, V., M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S.C. Shantz. 2005/12. Sizzle: A standards-based end-to-end security architecture for the embedded Internet. Pervasive and Mobile Computing 1, no. 4: 425-445.

Han, C., R. Kumar, R. Shea, E. Kohler, and M. Srivastava. 2005. A dynamic operating system for sensor nodes. In Proceedings of the 3rd international conference on mobile systems, applications, and services, applications, and services.

Hansen, M.S., and S. Støa. 2006. Practical evaluation of IEEE 802.15.4/ZigBee medical sensor networks. Ph.D. diss., Norwegian University of Science and Technology.

Heidemann, J., N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. 2001. Effects of details in wireless network simulation. In Proceedings of the SCS communication networks and distributed systems modeling and simulation conference.

Hui, J., P. Levis and D. Moss. TinyOS 802.15.4 frames. 2008/06/20 2008. Internet on-line. Available from <http://tinyos.net/tinyos-2.x/doc/txt/tep125.txt>. [03/29, 2009].

IEEE Computer Society. Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs). 7 June 2006 2006. [03/10, 2009].

Jones, W. D. 2006. No Strings Attached. IEEE Spectrum 2006, no. April (April 2006) : 16-16-18.

Jones, W. D. 2004. Ultrawide gap on ultrawideband. IEEE Spectrum 41, no. 1 (Jan 2004) : 30.

Karlof, C., N. Sastry, and D. Wagner. 2004. TinySec: A link layer security architecture for wireless sensor networks. In Proceedings of the 2nd international conference on embedded networked sensor systems 162-175. Baltimore, MD, USA: ACM Press.

Kim, Y., R. Evans, and W. Iversen. 2006. Instrumentation and control for wireless sensor network for automated irrigation. In Proceedings of the american society of agricultural and biological engineers international (ASABE).

www.manaraa.com

Koh, B. K., P. Kong. 2006. Performance Study on ZigBee-Based Wireless Personal Area Networks for Real-Time Health Monitoring. ETRI Journal 28, no. 4: 537-540.

Koubaa, A., M. Alves. 2005. A two-tired architecture for real-time communications in large-scale wireless sensor networks: Research challenges. In In proc. of 17th euromicro conference on real-time systems (ECRTS'05), WiP session.

Kumagai, J., S. Cherry. 2004. Sensors & Sensibility. IEEE Spectrum 41, no. 7: 22-28.

Kuo, C., M. Luk, R. Negi, and A. Perrig. 2007. Message-in-a-bottle: User-friendly and secure key deployment for sensor nodes. In Proceedings on the 5th international conference on embedded networked sensor systems233-246. New York, NY, USA: ACM.

Leal, J., A. Cunha, M. Alves, and A. Koubaa. 2007. On a IEEE 802.15.4/ZigBee to IEEE 802.11 gateway for the ART-WiSe architecture. In Emerging technologies & factory automation, 2007. ETFA. IEEE conference on, edited by A. Cunha, 1388-1391.

Leavitt, N. 2007. For Wireless USB, The Future Starts Now. Computer 40, no. 7: 14-16.

Levis, P., N. Lee, M. Welsh, and D. Culler. 2003. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In Proceedings of the 1st international conference on embedded networked sensor systems126-137.

Liu, D., and P. Ning. 2007. Security for wireless sensor networks. New York, NY: Springer.

Luk, M., G. Mezzour, A. Perrig, and V. Gilgor. 2007. MiniSec: A secure sensor network communication architecture. In Proceedings of the 6th international conference on information processing in sensor networks479-488.

Luk, M., A. Perrig, and B. Whillock. 2006. Seven cardinal properties of sensor network broadcast authentication. In Proceedings of the fourth ACM workshop on security of ad hoc and sensor networks 147-156ACM Press.

Marandin, D. Simulation of IEEE 802.15.4/ZigBee with network simulator-2 (ns-2).Internet on-line. Available from <http://www.ifn.et.tu-dresden.de/~marandin/ZigBee/ZigBeeSimulation.html>. [11/20, 2007].

marvelouskobe. Spider-spots. July 14, 2008 2008. Internet on-line. Available from <http://www.youtube.com/watch?v=NskP-4AjkRQ>. [31/12, 2008].

McVoy, L.. LMbench - tools for performance analysis. 98/06/26 1998. Internet on-line. Available from <http://www.bitmover.com/lmbench/>. [05/08, 2007].

Melkonian, M., J. Wu. 2005. ZWOS - A lightweight operating system for lightweight embedded wireless devices. In TENCON 2005 2005 IEEE region 101-6.

www.manaraa.com

Microchip Technology Inc. PICDEM Z demonstration kit. 12/8/2008 2008. Internet on-line. Available from <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en021925&part=DM163027-2>. [03/29, 2009].

Microchip Technology Inc. MiWi. 2006. Internet on-line. Available from <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2113&param=en520414>. [2/22, 2007].

Mumford, R. 2006. Chipcon Acquired by Texas Instruments. Microwave Journal 49, no. 2 (February 2006) : 37.

Nokia Corporation. Wibree. 2007. Internet on-line. Available from <http://www.wibree.com/>. [06/12, 2007].

Nokia Corporation. Wibree forum merges with bluetooth SIG. June 12, 2007 2007. Internet on-line. Available from <http://press.nokia.com/PR/200706/1132534_5.html>. [06/12, 2007].

PacketHop Inc. PacketHop announces availability of draft IEEE 802.11s mesh product. April 28, 2008 2008. Internet on-line. Available from <http://www.packethop.com/company/news_events/press_releases/2008/042708.php>. [05/08, 2008].

Park, C., P. H. Chou. 2006. EmPro: An Environment/Energy emulation and profiling platform for wireless sensor networks. In Annual IEEE communications society on sensor and ad hoc communications and networks158-167.

Perrig, A., R. Szewcyk, J. D. Tygar, V. Wen, and D. E. Culler. 2002. SPINS: Security Protocols for Sensor Networks. Wireless Networks 8, no. 5 (September 2002) : 521-534.

Piotrowski, K., P. Langendorfer, and Peter Steffen. 2006. How public key cryptography influences wireless sensor node lifetime. In Proceedings of the fourth ACM workshop on security of ad hoc and sensor networks.

Polley, J., D. Blazakis, J. McGee, D. Rusk, J. S. Baras, and M. Karir. 2004. ATEMU: A fine-grained sensor network simulator. In First annual IEEE communications society conference on sensor and ad hoc communications145-152.

Pulse~Link. Pulse~Link introduces first ultra wideband whole-home interactive HD solutaion at CES-2008. January 4, 2007 2007. Internet on-line. Available from <http://www.pulselink.net/press/pr-jan04-2007.htm>. [03/06, 2007].

Ran, P., S. Mao-heng, and Z. You-min. 2006. ZigBee routing selection strategy based on data services and energy-balanced ZigBee routing. In 400-404.

www.manaraa.com

Research Studios Austria. ZigBee for linux :: CC2420 gumstix board. 10/08/2007 2007. Internet on-line. Available from <http://pervasive.researchstudio.at/portal/node/42>. [03/29, 2009].

Schilit, B. N., U. Sengupta. 2004. Device Ensembles. Computer 37, no. 12 (December 2004) : 56-64.

Schneier, B. 1996. Applied cryptography: Protocols, algorithms, and source code in C. John Wiley & Sons, Inc.

Sharp, C., M. Turon and D. Gay. Timers. 05-Sep-2007 2007. Internet on-line. Available from <http://www.tinyos.net/tinyos-2.x/doc/html/tep102.html>. [03/29, 2009].

Simon, D., C. Cifuentes, D. Cleal, J. Daniels, and D. White. 2006. Java™ on the bare metal of wireless sensor devices: the squawk Java virtual machine. In Proceedings of the 2nd international conference on Virtual execution environments : 78-88.

SmartLabs Inc. Insteon: A smartlabs technology. 2007. Internet on-line. Available from <http://www.insteon.net/>. [2/22, 2007].

Song, L., D. Hatzinako, and X. Wang. 2008. Wireless mesh infrastructure for ubiquitous voice and video. In Consumer communications and networking conference, 2008. CCNC 2008. 5th IEEE1267-1268.

Stajano, F. 2002. Security for ubiquitous computing. West Sussex, England: John Wiley & Sons, LTD.

Stajano, F, R. Anderson. 2002. The Resurrecting Duckling: Security Issues for Ubiquitous Computing. Computer 35, no. 4, Part Supplement (Apr 2002) : 22-26.

Sukumaran, A. 2006. Park view hotel.

Sun Microsystems Inc. Project sun SPOT products. 2009. Internet on-line. Available from <http://sunspotworld.com/products/>. [03/30, 2009].

Sun Microsystems Inc. SunSPOTWorld educational sales. 2009. Internet on-line. Available from <http://sunspotworld.com/products/edusales/>. [03/30, 2009].

TinyOS. TinyOS FAQ. 2009. Internet on-line. Available from <http://www.tinyos.net/faq.html#SEC-77>. [03/28, 2009].

UCLA Compilers Group. Avrora - the AVR simulation and analysis framework. Nov 25, 2007 2007. Internet on-line. Available from <http://compilers.cs.ucla.edu.erl.lib.byu.edu/avrora/>. [10/25, 2007].

Weiser, M. 1991. The Computer for the Twenty-first Centry. Scientific American 265, no. 3 (Sept 1991) : 94-104.

www.manaraa.com

Wen, Y. Distributed simulation of heterogeneous sensor network. Nov. 28, 2005 2006. Internet on-line. Available from <http://www.cs.ucsb.edu/~wenye/disens.html>. [10/25, 2007].

Wolfe, J. NICTA-UNSW robot clarinet. 2008. Internet on-line. Available from <http://www.phys.unsw.edu.au.erl.lib.byu.edu/jw/clarinetrobot.html>. [03/29, 2009].

Zhu, S., S. Setia, and S. Jajodia. 2003. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In Proceedings of the 10th ACM conference on computer and communications security62-72ACM Press.

Zhu, Y.W., X.X. Zhong, and J.F. Shi. 2006. The Design of Wireless Sensor Network System Based on ZigBee Technology for Greenhouse. Journal of Physics: Conference Series 48: 1195-1119.

ZigBee Alliance. Our members. 2008. Internet on-line. Available from <http://www.zigbee.org/en/about/members.asp>. [05/08, 2008].

ZigBee Alliance. FAQs. 2007. Internet on-line. Available from <http://www.zigbee.org/en/about/faq.asp>. [05/15, 2007].

ZigBee Alliance. 2007. ZigBee Specification.

ZigBee Alliance. 2006. ZigBee Specification.

ZigBee Alliance. 2004. ZigBee Specification.

Z-Wave Alliance. 2007. Internet on-line. Available from <http://www.z-wavealliance.org/>. [1/22, 2008].

www.manaraa.com