This is to certify that the
thesis entitled

An Energy Efficient Link-Layer Security Protocol for Wireless Sensor Networks

presented by

Leonard E. Lightfoot

has been accepted towards fulfillment
of the requirements for the

M.S. degree in Electrical Engineering

Tongtong Li
Major Professor's Signature

12/8/2006
Date

AN ENERGY EFFICIENT LINK-LAYER SECURITY PROTOCOL FOR WIRELESS SENSOR NETWORKS

By

Leonard E. Lightfoot

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical & Computer Engineering

2006

ABSTRACT

AN ENERGY EFFICIENT LINK-LAYER SECURITY PROTOCOL FOR WIRELESS SENSOR NETWORKS

By

Leonard E. Lightfoot

In recent years, wireless sensor networks (WSNs) have found use in a variety of different applications including environmental monitoring, battlefield strategy planning, health monitoring, and so forth. With many of the applications involving communication of highly sensitive data, security becomes a primary concern. Generally, features such as data integrity, data authentication and information confidentiality are required for secure communications. However, incorporating these functions in WSNs is challenging due to the specific constraints such as limited memory and restricted energy supply. These constraints prohibit conventional security techniques from being directly applied to WSNs. As a result, new energy efficient protocol designs are highly desired for WSNs.

In this thesis, we investigate the importance of the link-layer security service in WSNs and propose an energy efficient link-layer security protocol (LLSP) to reduce the energy consumption in the network. The LLSP protocol is based on the security protocol for TinyOS applications called TinySec, which provides node authentication, message confidentiality and access control. In addition to providing the security services of TinySec, LLSP provides replay protection while reducing the security overhead per packet by 17%. Throughout this thesis we analyze and compare the performance as well as the energy consumption of TinySec and LLSP security protocols. Furthermore, we investigate the throughput of the system over both error-free and lossy channels using the LLSP protocol. The simulation results demonstrate that LLSP outperforms TinySec in terms of energy reduction and throughput performance.

To my parents Betty and Leroy Lightfoot, and my brother, sister and nephew.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my appreciation to my advisor, Dr. Tongtong Li, for her continuous support, guidance and encouragement throughout the years. She has helped me in every aspect, from providing advice on research to personal development and growth. She is more than an advisor, she is a friend.

I want to thank Dr. Percy Pierre and Dr. Jian Ren from the Department of Electrical and Computer Engineering for serving on my committee. I am deeply indebted to them for their support, either in the classroom or in all thoughtful correspondences. I would also like to thank Dr. Barbara O'Kelly for her valuable advice and for making my transition to graduate school a smooth one.

I am grateful to all my friends who have made my life at Michigan State University an enjoyable experience. Thanks to my friends from the Sloan program and NSBE organization for treating me like family. I would also like to thank all my friends around the nation, Jamin Butler, DeAndre Cole, Domonic Young and Christopher Bolden, for all the fun times and encouraging conversations. I would like to send a special thank you to my lab mates Dr. Weiguo Liang, Qi Ling and Dr. Huahui Wang for our valuable discussions on research. Huahui, thanks again, couldn't have done it without you.

Lastly, I would like to thank my parents, my brother, my sisters, my nephew, my nieces and my extended family for their love and constant support. Special thanks goes to my love, my girlfriend, Tinesha Mitchell, who has supported me even when I have doubted myself. She is the true definition of a friend, one that remains through thick and thin and I will always love her for that.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

"Images in this thesis are presented in color"

CHAPTER 1

Introduction

The introduction chapter discusses the applications, structure, major challenges and security requirements of wireless sensor networks (WSNs). In addition, this chapter reviews some of the existing security protocols and briefly describes the proposed energy efficient link-layer security protocol.

1.1 Security for Wireless Sensor Network

1.1.1 Wireless Sensor Networks

Wireless sensor networks (WSNs) are defined as networks consisting of independent, collaborating nodes that can sense, process and exchange data as well as act upon the data content [2]. Independently each node is limited in its capability, but jointly the data-centric network can deliver time sensitive information to different destinations. Typical WSNs are ad-hoc networks and are completely un-tethered from a physical infrastructure. These networks are characterized by dynamic, unpredictable, random and multi-hop topologies. With the many advances in technology, WSNs often extend rather than replace wired networks. As a result, more civilian applications have been observed. This section provides information about applications, components and routing protocols of WSNs.

A. Applications

WSNs have increased in popularity in many applications because manufacturing of the small and low cost sensors are more technically and economically feasible. It is

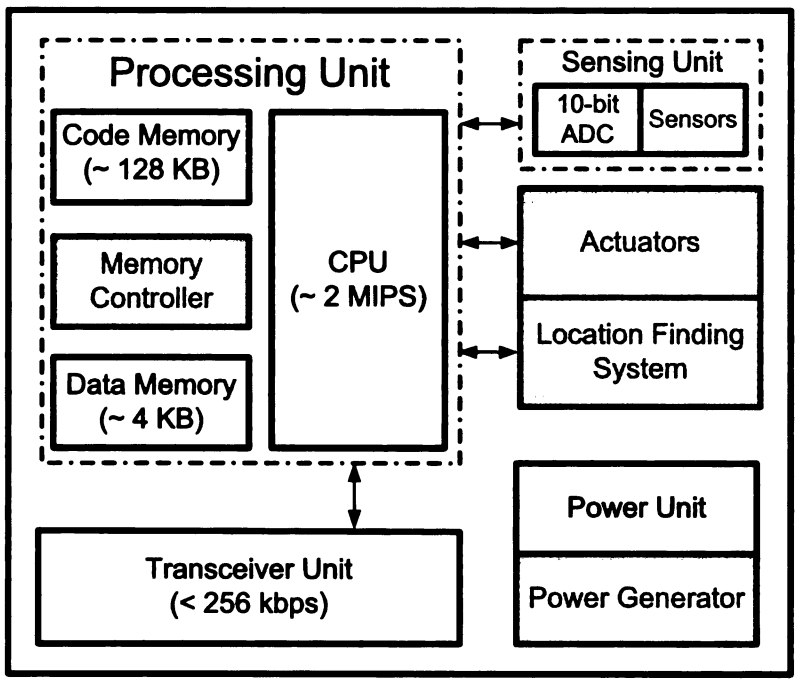


Figure 1.1. Architecture of a sensor node [2]

expected that in the future thousands of network nodes will be able to operate unattended sensing tasks. These networks are likely to be widely deployed in a vast variety of environments for commercial, civil and military use. Commercially, these nodes can be used to provide information about traffic jams and the speed and density of traffic. The civil applications include environmental and habitat monitoring. Environmental monitoring involves monitoring the air, soil and water conditions, whereas habitat monitoring entails monitoring plant and animal species population and behavior [4]. Military applications range from large-scale acoustic surveillance systems for ocean surveillance to small networks of unattended ground sensors for ground target detection [5].

B. Sensor Components

Sensors are composed of four major components: a sensing unit, processing unit, transceiver and a power unit. Each component is described in the subsections below and depicted in Figure 1.1.

- **Power Unit:** The power unit provides power to the node so the collective components operate effectively. Typically, batteries are used in sensors to provide power to individual sensor components and are either non-rechargeable or rechargeable. Non-rechargeable batteries are seldom used in current sensor designs and are mostly used in harsh environments that prevent the replacing or recharging of batteries. Current sensor designs include energy restoration via solar or vibration energy [6–8]. However, the sensor's lifespan is primarily extended by energy preservation schemes such as completely shutting down inessential components rather than putting them into idle mode. As shown in Figure 1.2, idle mode operation consumes approximately the same amount of energy as the receiving mode operation.

- **Sensing Unit:** The sensing unit contains numerous sensors to measure physical data from a targeted area. The physical data collected is generally a continuous analog signal which is digitalized with an analog-to-digital converter (ADC). The digitalized data is later processed by the processing unit for analysis.

- **Processing Unit:** The processing unit consists of a central processing unit (CPU), storage devices and an optional memory controller [2]. This unit is responsible for controlling the sensors and executing the communication protocols.

- **Transceiver:** The main function of the transceiver is to communicate with neighboring nodes and the outside world. Communication is achieved either via optical (laser), infrared or by radio frequency (RF). Optical communication consumes the least amount of energy, but requires a line of sight and is sensitive to atmospheric conditions [7]. Infrared is similar to laser, but is limited in its broadcasting capacity. The widely used RF, consumes the most energy in relation to its counterparts, yet it is preferred because of its broadcasting capabilities.

  An interesting fact is that communication remains one of the most energy consuming operations. As shown in Figure 1.2, transmitting and receiving opera-

tions consume the most energy compared to energy consumption of processing. To save energy it is desirable to transmit and receive less overheads, which can be done under the security framework investigated in this thesis.

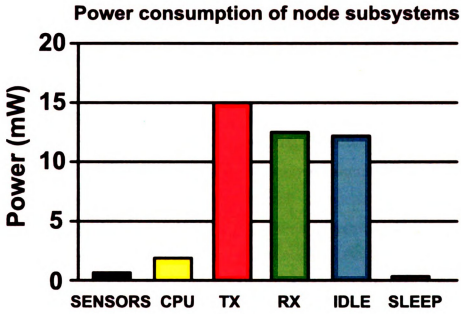


Figure 1.2. Power Consumption of sensor node subsystems [3]

C. Routing Structure

The routing protocol is an essential component of the WSN. It defines the routing format of the network. Depending on the application and network architecture, the design of the routing protocol can vary. In [9], the authors classify the routing protocols into three categorizes based on the underlying network structure: *flat*, *hierarchical* and *location-base*.

Flat routing protocols equally distribute the energy, process capability, memory and sensing task among all sensors in the network. The vision of a large scale WSN, makes it unfeasible to assign a global identifier to each node. For this reason, routing has moved to data-centric routing, where the base station sends queries to certain regions and waits for data from the sensors located in the selected regions [9]. *Sen-*

sor Protocols for Information via Negotiation [10,11] and *Directed Diffusion* [12], are among the early works of data-centric routing protocols. Both protocols seek to conserve energy, reduce redundancy and minimize the number of overhead transmissions in the network. These two protocols have motivated and pioneered the design of many other routing protocols.

With hierarchical routing protocols, a fraction of the nodes in the network are more powerful than others in terms of processing capability, energy and memory. These nodes are called "cluster heads" and recruit sensor nodes in the surrounding area as "cluster members". Only the cluster members collect sensed data and only cluster heads route data to a base station [13]. The two layer architecture of hierarchical routing is designed to reduce the energy consumption, distribute operation tasks and prolong the lifetime of the network. Protocols such as Low Energy Adaptive Clustering Hierarchy (LEACH) [14], Power-Efficient Gathering in Sensor Information Systems (PEGASIS) [15], and Threshold-Sensitive Energy Efficient Protocols (TEEN) [16] are a few hierarchical based routing protocols with a shared goal of preserving energy. Each of the three protocols have developed a unique approach to minimize energy consumption; LEACH utilizes a randomized rotation of cluster heads to evenly distribute the energy load among the sensors in the network; PEGASIS, an enhancement of LEACH, uses a round robin technique among the sensor nodes to communicate with cluster heads to uniformly spread the power dissipation over all nodes; TEEN protocol, geared toward time-critical applications, uses a threshold technique to reduce the number of data transmissions that state little or no change in the sensed attribute.

In the location based protocol, sensor nodes are addressed by means of their location. This protocol is primarily designed for mobile ad-hoc networks but may be applied to sensor networks [9]. The distance between neighboring nodes can be determined by measuring the incoming signal strength. The location is ascertained by communicating with a satellite using GPS if nodes are equipped with a small low-power GPS receiver. The Geographic Adaptive Fidelity (GAF) [17], is a location based routing protocol that utilizes the GPS technology to associate sensor nodes to

a point in the virtual grid. On the contrary, the SPAN [18] protocol uses a few sensor nodes as coordinators based on their positions.

1.1.2 Security Requirements

Incorporating a security protocol in WSNs is a challenging, but necessary feature to include in the network design. An adequate security service can prevent malicious power consumption attacks, ensure effective access control, and provide information confidentiality [19]. In this section we review four basic security requirements for a superior link-layer security protocol: access control, message authentication, message confidentially and message replay protection.

A. Message Confidentiality

The goal of message confidentiality is to keep information secret from unauthorized parties. Typically, confidentiality is achieved with encryption. A proficient encryption scheme prevents an adversary from recovering an encrypted message and prevents an adversary from learning partial information about the encrypted message. This strong property of encryption is known as semantic security [20], which implies it is infeasible for an adversary to derive significant information about a message plaintext when given only its ciphertext and secret encryption key.

Due to the broadcast nature of WSNs, data encryption is extremely important. Wireless routed data makes it easier for eavesdroppers and adversaries to capture messages. Ideally, a strong encryption scheme is sought for wirelessly transmitted data, but generally, the stronger the encryption scheme, the more energy inefficient it becomes. This is due to the lengthy key storage and extra computational processing required for strong encryption algorithms.

B. Message Authentication and Access Control

Message authentication ensures a message received has not been modified or altered during transit. If the message is altered during transit, message authentication will

allow the receiver to detect any altering or tampering. Message authentication is generally achieved by appending a message authentication code (MAC) to each transmitted packet. A MAC is essentially a one-way hash function with a secondary secret key input. A one-way hash function maps bit-strings of arbitrary finite length into strings of fixed length, such that it is computationally infeasible to find any input which hashes to any pre-specified output. A hash function also makes it is computationally infeasible to find any second input which has the same output as any specified input [21].

Due to the MAC's cryptographic computation, it can also be used to provide access control in WSNs. Access control imply that the link-layer protocol prevent unauthorized parties from participating in the network. A MAC allows legitimate nodes the ability to detect and reject messages from unauthorized nodes.

C. Replay Protection

Replay protection prevents messages from being re-sent to an authorized receiver. It ensures that all messages received by the receiver are fresh and has never been transmitted previously. If an adversary eavesdrops on a message between two authorized nodes and later replays the message to the receiver, the receiver will accept the message because the message is originally from an authorized sender. Due to the limited amount of state that each recipient can keep, replay protection is difficult to ensure.

A typical defense is to include a monotonically increasing counter with every message and reject messages with old counter values [19]. However, this method requires each recipient to maintain a table of the last value from every sender it receives. For RAM-constrained sensor nodes, this method is problematic for even modestly sized networks. Also, transmitting the counter value for each message packet is energy inefficient because data communication consumes a large amount of energy.

1.1.3 Major Challenges on Sensor Network Security

The structure, design and vision of WSNs, introduces many challenges to implement security protocols. Typically, WSNs are constrained in energy and bandwidth since the sensors are small, low-power, and low-cost devices.

A. Power Constraint

Power is the biggest constraint against implementing security for WSNs. It is envisioned that these sensors once deployed will operate unattended in hostile environments, thus making sensor replacement unfeasible. In an intent to extend the longevity of the sensor network, security services are frequently disregarded due to the limited energy. Consequently, this leaves WSNs vulnerable to security attacks, which could consume excessively more battery power and shorten the longevity of the WSN. In the worst case, an adversary can take control of the sensor nodes and compromise the cryptographic keys to reprogram the sensor nodes.

As applications for WSNs begin to operate on sensitive data, security becomes an upmost importance. However, incorporating security to WSNs is costly. The additional energy consumed by sensors due to security is related to the processing required for cryptographic primitives (e.g., encryption, decryption, message authentication, and message verification), the energy required to transmit the security overhead (e.g., initialization vectors needed for encryption/decryption and message authentication codes), and the energy required to store security parameters in a secure manner (e.g., cryptographic keys) [22].

Finally, a major challenge in designing security protocol for WSNs lie in the fact that traditional security techniques cannot be applied directly in WSNs [23]. Traditional security algorithms require lengthy cryptographic keys and message authentication codes (MACs) that consumes severe processing power and designed for powerful workstations. Furthermore, the demand of ad-hoc routing protocols complicates the system design. Hence, to design secured protocols in WSNs the following elements are taken into consideration: a secure access control protocol, simple but effective

data encryption/decryption algorithm, and efficient and practical key management scheme [19].

B. Limited Memory and Storage Space

The nodes used in WSNs are tiny devices with only a small amount of memory and storage space for program code. If a security feature is incorporated into the design of these nodes, it is a necessity that the code size of the security algorithm is minimal. A common sensor used for these networks is the Mica2 node with 7.3MHz ATmegal128L processor, 128KB of code memory, 512KB flash storage, 4KB of data memory and a Chipcon CC1000 radio capable of transmitting at 38.4kbps with an outdoor transmission range of approximately 300m [24, 25].

C. Unattend Operation

WSNs are planned to operate unattended sensing tasks in hostile and/or harsh environments, which leaves the network vulnerable to various security attacks. An adversary can reverse-engineer, modify and abuse the network if a node is captured. For example, an adversary can acquire detailed knowledge of sensors' task and objective, modify the sensors with malicious code and produce and deploy multiple copies of the manipulated sensor device into the network. Therefore it is vital to make sensor devices *tamper-proof*, which is discussed in [26].

D. Unreliable Channel

Wireless channels by nature depend on a broadcast medium, which is inherently unreliable. Due to the higher error rates of a wireless channel, packets have a higher probability to become corrupt or lost during transit. Furthermore, the multi-hop infrastructure of these networks introduce node synchronization problems, which can be critical to sensor security mechanism such as cryptographic key distribution [27].

In addition to the wireless channel being unreliable, it is more vulnerable to security attacks than its wired counterpart due to the lack of physical boundary. An adversary with an appropriate transceiver can eavesdrop, intercept, inject and alter

the transmitted data. As a result, a security service is a necessity to ensure information confidentiality and effective access control in WSNs.

1.2 Related Work

1.2.1 Existing Security Protocols

In this section we review some representative security protocols in literature and discuss their advantages, disadvantages and limitations.

A. SPINS

The research topic of security protocols for WSNs is a fairly new topic. The research group at the University of California Berkeley is perhaps one of the first to publish a secure protocol specifically design for the inexpensive and low-power devices. In their *"Secure Protocols for Sensor Networks (SPINS)"* paper [23], they introduce the secure network encryption protocol (SNEP). This protocol is designed and developed to provide confidentiality, data authentication and data freshness. The communication overhead is 8 bytes per message and like many cryptographic protocols it uses a counter to provide replay protection. The communicating parties share the counter and increment it after each block, so the counter does not need to be sent with the message [23]. Data authentication is provided by the use of a MAC and data confidentiality is provided via block cipher RC5 [28].

In [29], Luo and Zheng refutes SPINS's recommendation of the RC5 block cipher. The claim is that RC5 operates on 32-bit words which is too expensive for a 8-bit Advanced Virtual RISC (AVR) architecture [29]. Luo and Zheng finds that smaller alternatives such as the Tiny Encryption Algorithm (TEA) [30] algorithm is more suitable for these low-end devices and that TEA is the perfect encryption algorithm to maximize speed and minimize memory and energy usage. However, the security for these smaller alternatives such as TEA and TREYFER [31], have yet to be thoroughly analyzed; therefore an encryption algorithm with the ability to attain higher security

is recommended.

The design of the SNEP protocol is superior considering it is one of the first security protocols proposed. However, the SNEP protocol is not implemented nor fully specified. Researchers have proven that the RC5 block cipher is too expensive for these low-end devices. In all, the SNEP protocol provides a solid infrastructure to designing efficient and secure protocols for WSNs, and motivated the design of many other protocols including the security protocol proposed in this thesis.

B. IEEE 802.11

The IEEE 802.11 standard for wireless networks includes a wired equivalent privacy (WEP) scheme to provide link-layer protection against eavesdropping and other attacks. The WEP protocol uses the stream cipher RC4 for confidentiality, cyclic redundancy check (CRC) for integrity protection and a 3-byte initialization vector (IV). Many researchers [32–34] have found WEP to be thoroughly flawed and should not be counted on to provide strong link-layer security.

The first major flaw is the RC4 stream cipher as their encryption scheme. A well known drawback of any stream cipher is that encrypting two message under the same IV and key can reveal information about both messages. The second major flaw is the use of a CRC checksum for message authentication. CRC's are designed to detect random errors in the message. However, CRC is not resilient against malicious attacks. In [32], the authors demonstrate this vulnerability of CRC and claim that it is exacerbated by the fact that the message payload is encrypted using a stream cipher. The third major flaw is the use of short IVs to diversify RC4's keys. The WEP protocol recommends but does not require that the IVs to be change after every packet. In addition, the WEP protocol does not mention anything about how to select the IVs. The fact that the IV is only 3 bytes in length nearly guarantee that the same IV will be used for multiple messages.

Recently, IEEE has improved on the WEP flaws and adopted new standards such as the Temporal Key Integrity Protocol (TKIP) [35] and Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP) [36]. These

improved standards are designed with stronger message authentication and better IV mixing with keys. However, these improved standards require a huge overhead per packet. The CCMP uses the Advance Encryption Standard (AES) in Counter with CBC-MAC (CCM) mode, 48-bit IVs and a lengthy 64-bit MAC. CCMP is well designed but the overhead is too large for sensor networks.

C. TinySec Protocol

TinySec [37] is the standard security protocol for TinyOS [38], an operating system for sensor nodes. The TinySec protocol provides two options for security. The first option only provides authentication for each message packet. The second option provides authentication and encryption for each message packet. Many applications require authentication and encryption for secure communication; thus only the latter option is exploited. The latter security option requires a security overhead of 12 bytes per message transmission and uses a MAC to ensure message integrity. Unlike the SNEP protocol, the counter value is transmitted with each message and not maintained by each recipient. Transmitting the counter value during transmission has its tradeoff: The recipient does not have to maintain a counter value for each sender which will yield less consumption of process and memory resources. However, the sender and recipient will consume more energy by transmitting the counter bytes because communication operations consume the most energy.

The link-layer security protocol proposed in this thesis is based on the TinySec security protocol, with similar features but important differences. One of the key differences between the TinySec security protocol and the proposed protocol is the disregarding of the counter bytes during transmission. A brief summary of the proposed security protocol is explained in Section 1.3, whereas a detailed explanation can be found in Chapter 3.

1.2.2 Other Related Protocols

In this section we review some other related protocols in WSNs and discuss their cryptographic measures.

Compared to its wired counterpart, wireless security protocols has a much wider range of applications to secure. For example, the cellular technologies uses protocols such as CDPD [39] and GSM [40]. The wireless local area network (WLAN) uses protocols such as IEEE 802.11 [36] and wireless personal area network uses protocols such as Bluetooth [41]. Many of these protocols are designed to secure network access domain. In other words, these protocols secure the link between a wireless client and the access point base station. However, many researchers [32,42–44] have found these protocols unsecured due to their ease of breaking into.

Many security protocols utilize cryptographic algorithms such as asymmetric (public-key ciphers) or symmetric (private-key ciphers) encryption as the building blocks to provide a secure communication. In Patel's and Crowcroft's paper [45] they focus on security solutions for mobile user devices. Their design is based on a asymmetric encryption algorithm which is proved to be too expensive for the wireless sensor environment. Likewise, the work presented in [46], uses asymmetric cryptography for authentication. Authors, Zhou and Hass propose a secure routing and secure key management service in an ad-hoc network using asymmetric cryptography [47]. Despite the many asymmetric cryptographic algorithms available, symmetric encryption algorithms are inherently well suited for these low-end devices due to their relative low overhead [23].

1.3 Proposed Link Layer Security Protocol

1.3.1 Motivation

The motivation to develop an energy efficient link-layer security protocol is threefold: (i) secure communication is always desired, (ii) prevent energy consuming attacks by using link-layer security mechanisms and (iii) reduce energy consumption by mini-

mizing the security overhead.

Firstly, data security is a major concern for users when transmitting data through a physical or wireless medium; therefore securing data transmission is of utmost importance. A major goal of WSNs is to provide protection for the transmittance of highly sensitive data. Security issues can hinder the wide-spread deployment of sensors. In addition, as mentioned in Section 1.1.3, wireless sensors have several inherent limitations in their design. Incorporating a security protocol despite these limitations would be quite challenging because security services will require additional communication overhead which consumes more of the limited energy.

Secondly, in conventional networks, security services (e.g., message authentication and confidentiality) are usually achieved by an end-to-end security mechanisms such as SSH [48], SSL [49], IPSec [50]. With this style of communication the intermediate routers do not access the message body and only view the message headers to relay the message. In contrast, WSNs communicate over a multi-hop topology and often consist of several sensors witnessing the same or correlated environmental events. As a result, neighboring sensors will transmit the same or correlated events, causing precious energy and bandwidth to be wasted. To minimize the number of redundant messages, sensor networks use in-network processing such as aggregation and duplicate elimination. Unlike the intermediate routers in end-to-end security mechanisms, in-network processing requires the intermediate routers to access, modify and suppress the contents of the message. To achieve node-to-node authentication, integrity and confidentiality a link-layer security architecture must be incorporated into the network design. In addition to providing node-to-node security services, a link-layer security protocol prevents denial of service attacks and detects unauthorized packets when they are first injected into the network versus an end-to-end security mechanism that allows unauthorized packets to propagate throughout the network.

Thirdly, it is widely known that communication is the dominant source in energy depletion for these sensor nodes. As shown in Figure 1.2, the transmitting and receiving operations are the leading energy consumers. Theoretically, the longer a communication message is, the more energy it consumes to transmit that message.

Therefore, to efficiently incorporate security services for WSNs, we need to minimize the number of bits transmitted per message. In other words, the security overhead needs to be minimal.

For the above reasons, in this thesis we propose an energy efficient link-layer security protocol (LLSP) which aims at minimizing security communication overhead while upholding the security requirements of WSNs.

1.3.2 Link Layer Security Protocol

The design of the LLSP protocol is based on the security protocol for TinyOS applications [1]. However, LLSP security protocol reduces the energy consumption by minimizing the security overhead for each message packet while maintaining security primitives such as message authentication, replay protection and message confidentiality. The LLSP security protocol only transmits the destination address, source address, active message handler and the packet length for each message transmission, which results in transmitting only 10 bytes of security overhead. Recall that the security overhead for TinySec security protocol is 12 bytes. The reduction in security overhead in the LLSP protocol is achieved by removing the counter value bytes during each message transmission. The counter value is maintained by a synchronous 4-byte counter (e.g., feedback shift register) between the sender and receiver pair. This allow advantages such as replay protection and the ability of the receiver to determine the number of lost messages packets based on the correct counter value. Furthermore, the MAC is not encrypted with the LLSP security protocol. This design enables the receiver to determine the authenticity of the message with minimum energy consumption since the decision can be made without requiring decryption of the message packet [19].

The LLSP protocol upholds the security requirements while theoretically reducing the energy cost of the security overhead by 17%. In an effort to prove the efficiency and security of LLSP, the energy consumption and throughput of the two protocols (LLSP and TinySec) are measured using the TinyOS simulator.

1.4 Thesis Outline

The reminder of this thesis is organized as follows. In Chapter 2, we formulate the problem solved in this thesis, and review the system model, power estimation strategies and operating systems for sensor networks. In Chapter 3, we present the energy efficient link layer security protocol, analyze and compare the energy consumption and throughput of the TinySec and LLSP security protocols, and validate that the LLSP security protocol is in fact energy efficient, significantly improves the security and is suitable for wireless sensor networks. Finally in Chapter 4, we conclude and discuss directions of future work.

CHAPTER 2

System Model and Problem Formulation

In this chapter we describe the system model that complement our proposed link-layer security protocol. In addition, we review power estimation strategies and operating systems for sensor networks.

2.1 System Model

Many researches have proven hierarchical networks are vital for efficient resource utilization and load balancing in large scale WSNs [51–56]. For these reasons, we choose the hierarchical network as the targeted architecture for the proposed link-layer security protocol. However, we must also mention that our proposed protocol is not limited to any topology. Our proposed protocol complements the hierarchical based network in achieving the perpetual goal of reducing energy consumption to increase the longevity of the network.

Hierarchical networks organize sensor nodes by clustering, an effective self-organization technique that can prolong the network lifetime. In a clustered network, a fraction of the sensor nodes are selected as cluster heads. A cluster head is responsible for (1) communicating with nodes within its cluster, typically via single hop or multi-hop, and (2) communicating with other cluster heads or with the observer(s) on behalf of its cluster. Figure 2.1 depicts the structure of a hierarchical WSN. The circles shown in the figure indicate the transmission range of each cluster head.

The network topology assumes a set of sensor nodes dispersed uniformly and

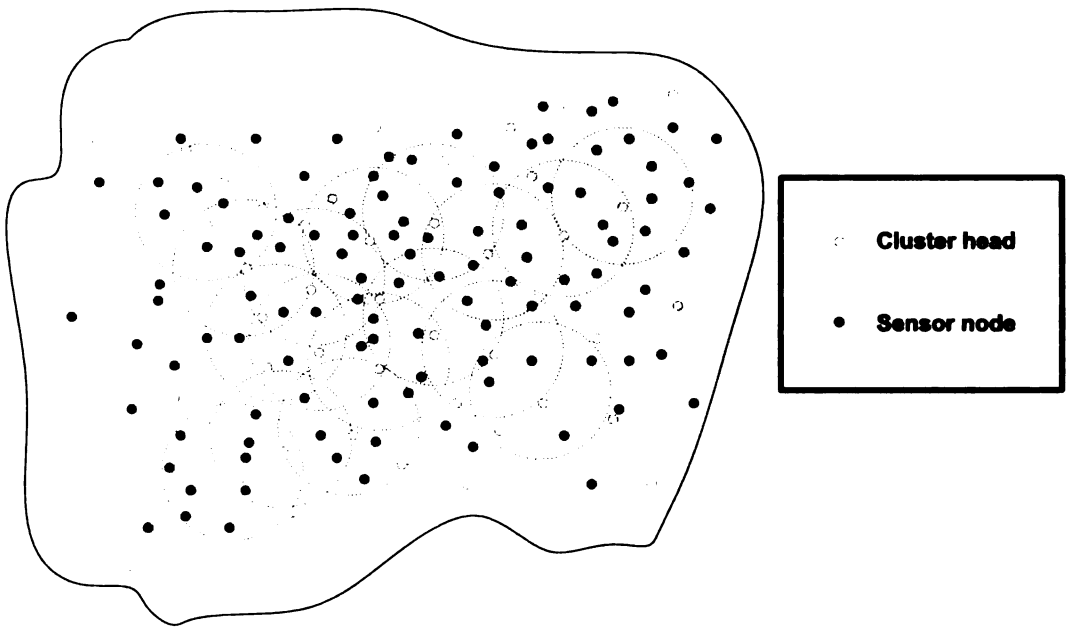


Figure 2.1. Typical wireless sensor network architecture

independently at random on a field. The network exhibits the following assumptions: (1) nodes are stationary, (2) nodes have an omni-directional antenna with a maximum transmission range r and symmetric link, i.e., if one node can hear from its neighbor, the neighbor can also hear from the node, (3) the cluster heads are more powerful than others in terms of processing capability, energy and memory, and (4) nodes are not location-aware and are left unattended after deployment.

The wireless channel assumed in this thesis is low-powered, and error-free with propagation effects neglected. Although no bits are corrupted due to error, two nodes can transmit at the same time. Every node within the transmission range will hear the overlap of the signals, which will cause a packet to corrupt. However, because of the error-free transmission, the probability of two nodes transmitting at the same time is very low due to the Carrier Sense Multiple Access Collision Avoidance (CSMA/CA) protocol. Furthermore, we assume each message packet has a fixed size.

A generic sensor node is also assumed in this thesis. A generic sensor node performs four fundamental operations: *sense* the environment, *transmit* and *receive* data,

and *listen* to the channel. Each fundamental operation consumes a different amount of energy. The *transmit* operation consumes the most energy whereas the sensing energy operation is neglected due to its minimal energy consumption. The *listen* operation, sometimes called the *idle* operation, requires the nodes' radio to remain active because the node does not know when it will receive a message from one of its neighbors. As a result, we assume that the *listen* operation consumes a constant amount of energy. Since each of the message packets are of fixed lengths, the energy cost to *receive* a message is constant as well.

2.2 Operating Systems for Sensor Network

Over the past few years there have been several operating systems developed specifically for wireless sensor networks. The operating systems are generally in charge of running the sensor's hardware, such as making sensor measurements, routing data, and dissipating the sensor's power. From these tasks, it is clear that the sensor's operation system is vital to the longevity of the network in every aspect. In this section we review some existing operating systems for sensor networks.

2.2.1 MagnetOS

The MagnetOS [57] developed at Cornell University is a distributed operating system for ad-hoc and sensor networks, whose goal is to enable power-aware, adaptive and easy-to-develop ad-hoc networking applications. MagnetOS achieves these goals by using a single system image of a unified virtual machine to applications over an ad-hoc collection of heterogeneous nodes. The developers of MagnetOS claim its operating system reduces energy consumption and increases system longevity by a factor of four to five, by automatically and transparently partitioning applications into components, and dynamically finding a placement of these components on the nodes within the ad-hoc network [57].

Increasing system longevity by a factor of four to five is an attractive feature, but MagnetOS only works on x86 laptops and StrongARM PDAs such as iPAQs, Axims

and Jornadas. Unfortunately, none of the listed platforms are ideal for wireless sensor networks.

2.2.2 SOS

SOS developed by researchers at University of California, Los Angeles (UCLA) is an operating system for mote-class wireless sensor networks [58]. SOS uses a common kernel that implements messaging, dynamic memory and module loading and unloading. Its unique feature is the ability to use dynamically loaded software modules to create a system supporting dynamic addition, modification, and removal of network services. Also, programs are written using standard C code and compilers. Debugging is supported via standard C code debuggers such as GDB. SOS supports a wide range of node platforms, such as many Crossbow Mica Motes, and Yale University's XYZ mote [59].

Although SOS supports the targeted Crossbow Mica Motes, it is not the desired operating system to support this thesis. The TinyOS operating system (Section 2.2.3) supports Crossbow Mica Motes and is fully equip with a network and power consumption simulator. The following subsection provides further details about the TinyOS operating system.

2.2.3 TinyOS

TinyOS, developed at the University of California, Berkeley, is a flexible, application-specific operating system for sensor networks [1]. It is designed to meet the sensor network challenges of limited resources, event-centric application and low-power operation. TinyOS is currently in its third generation involving several iterations of hardware, radio stacks, and programming tools.

TinyOS provides builtin interfaces, modules, and sensor-board specific configurations, which allow programmers to build programs as a set of modules, and perform program-specific tasks. TinyOS has three software components: *command*, *event* and *tasks*. *Commands* and *events* are mechanisms for inter-component communication,

while *tasks* are used to express intra-component concurrency [60]. A *command* is a request to a component to perform a service such as initiating the radio transceiver, whereas an *event* signals the completion of that service. Often, commands and event handlers are over burden with requests and may post a *task*, which is a function executed by the TinyOS scheduler at a later time. This feature allows commands and events to be responsive by deferring extensive computations to tasks. The basic execution model of tasks is to run to completion versus running indefinitely; this allows tasks to be much lighter-weight than threads.

The standard TinyOS operating system consists of a non-preemptive *First In First Out* (FIFO) task scheduler and numerous software components for radio communication, sensing, EEPROM access and other devices. A TinyOS application is assembled by linking multiple software components into an optimized binary, not in a binary kernel. Also, TinyOS is implemented in NesC [61], a C-based programming language. Over the past few years, the popularity of TinyOS has accelerated due to its support for several common sensor node platforms. These platforms include Mica, Mica2, MicaZ, Telos, MSP430 and the AVRMote.

A. TOSSIM and PowerTOSSIM

The TinyOS environment has a built in simulator called TOSSIM [1] and energy simulator called PowerTOSSIM [62]. In TOSSIM, the TinyOS application is complied into the TOSSIM framework, which runs on a personal computer (PC). TOSSIM captures the behavior and interactions of thousands of networked nodes at a very low level. The network is simulated at the bit level for each individual analog-to-digital conversion (ADC) capture and interrupt in the system. TOSSIM does not model the real world. Instead, it provides abstractions of certain real-world phenomena such as bit error. Due to TOSSIM flexibility, users can use tools outside the simulator itself to manipulate these abstractions to implement whatever models they want to use.

TOSSIM also provides a visualization tool called TinyViz. TinyViz is a Java-based graphical user interface for TOSSIM. The visual tool allows the user to debug, test and analyze algorithms in a controlled and repeatable environment while providing

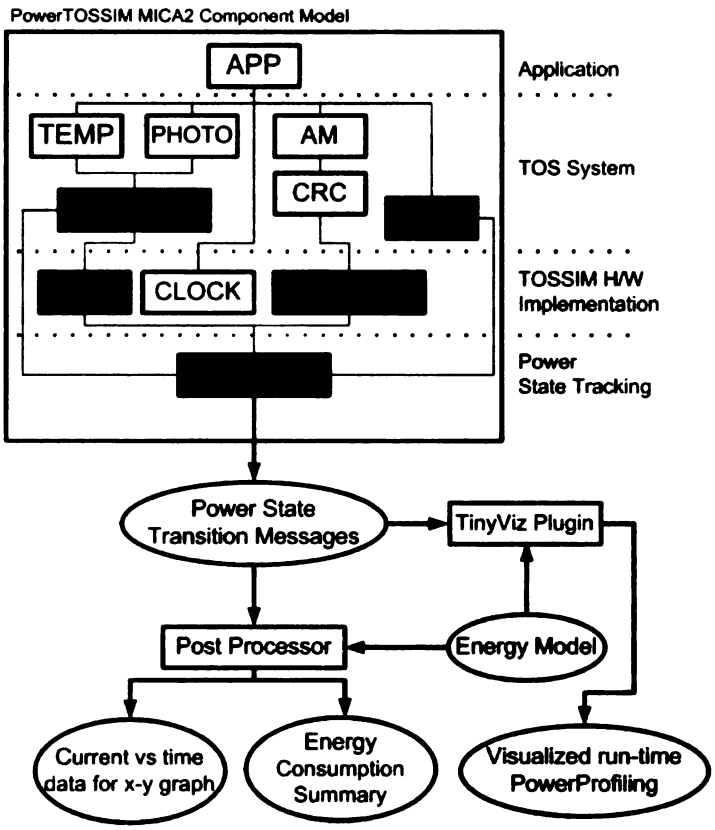


Figure 2.2. PowerTOSSIM architecture [1]

visual feedback on the simulation state and mechanisms for controlling the running simulation (e.g., modifying ADC reading and radio loss probabilities).

Although TOSSIM has many attractive features, it fails to provide an accurate power consumption model. TOSSIM is unable to model the CPU execution time; thus it cannot provide accurate information for calculating CPU energy consumption. However, TOSSIM's successor PowerTOSSIM is capable of calculating the CPU energy consumption and provides an accurate energy consumption.

PowerTOSSIM is an extension of TOSSIM and provides an accurate per node estimate of the power consumption. In PowerTOSSIM, specific hardware peripherals such as radio, EEPROM, and LEDs are instrumented to obtain a trace of each peripheral's activity during the simulation run time. However, estimating the CPU is more involved. Using trace files of the CPU is not feasible because PowerTOSSIM

runs the node software as a native binary on the host machine; therefore PowerTOSSIM has no information on the length of time that a given node spends using the CPU. CPU estimation is achieved by mapping the basic blocks executed by the simulation code to cycle counts in the corresponding node binary. Figure 2.2 is a visual of PowerTOSSIM's architecture. The simulated hardware components such as the radio, LEDs, and sensors, make calls to the PowerState module, which produces power state transitions messages for each component.

Mode	Current	Mode	Current
CPU		**Radio**	
Active	8.0 mA	RX	7.03 mA
Idle	3.2 mA	TX (-20 dBm)	3.72 mA
ADC Noise Reduce	1.0 mA	TX (-19 dBm)	5.21 mA
Power-down	103 µA	TX (-15 dBm)	5.37 mA
Power-save	110 µA	TX (-8 dBm)	6.47 mA
Standby	216 µA	TX (-5 dBm)	7.05 mA
Extended Standby	223 µA	TX (0 dBm)	8.47 mA
Internal Oscillator	0.93 mA	TX (+4 dBm)	11.57 mA
LEDs	2.2 mA	TX (+6 dBm)	13.77 mA
Sensor board	0.7 mA	TX (+8 dBm)	17.37 mA
EEPROM access		TX (+10 dBm)	21.48 mA
Read	6.2 mA		
Read Time	565 µs		
Write	18.4 mA		
Write Time	12.9 ms		

Table 2.1. Power model for the Mica2.

PowerTOSSIM uses a power model based on the Mica2 [24] sensor node platform. The developers of PowerTOSSIM generated and validated their resulting power model by developing a set of micro-benchmarks that exercise each component (radio, EEPROM, etc) independently [62]. As shown in Table 2.1, the energy model for the Mica2 sensor node encompass a wide range of current levels. The receive power is constant at 7.0mA but the choice of transmission power affects the current consumption considerably, from 3.7mA at -20dBm to 21.5mA at +10dBm.

To validate PowerTOSSIM's energy consumption values, several power traces of real TinyOS applications were measured. Table 2.2 shows the measured versus simu-

Benchmark	Simulated (mJ)	Measured (mJ)	Error(%)
Beacon	92.93	106.73	-12.9
Blink	940.26	931.72	0.85
BlinkTask	940.28	917.90	2.5
CntToLeds	1336.49	1330.00	0.45
CntToLedsAndRfm	2620.37	2562.00	2.3
CntToRfm	2028.09	1985.00	2.1
Oscilloscope	867.94	801.60	8.3
OscilloscopeRF	2136.45	2021.90	5.7
Sense	865.59	900.72	-3.8
SenseLightToLog	2133.89	2005.26	6.4
SenseTask	865.62	944.74	-8.3
SenseToLeds	868.70	977.73	-11.1
SenseToRfm	2152.27	2059.16	4.5
Average			**4.7**

Table 2.2. Measured versus simulated energy for various TinyOS applications [1]

lated energy for standard TinyOS distributed applications. Many of the applications perform some combination of sensing, blinking LEDs, radio transmission and/or data recording in the EEPROM. Each application was executed for 60 real or simulated seconds. As shown in the table, PowerTOSSIM is accurate, with an average error of only 4.7% compared to the actual node. The developers state the error maybe due to voltage fluctuations, noise, and rounding error in the experimental setup.

B. TinySec

TinyOS also has a standard security protocol call TinySec [37]. The TinySec protocol provides two options for security: *authentication only* and *authentication and encryption.* The authentication only security option authenticates the entire packet with a message authentication code, but the data payload is not encrypted. The authentication only security option provides weak security, but is suitable for applications such as a burglar alarm systems that transmits intrusion signals. Maintaining the secrecy of these intrusion signals is unnecessary and only increases latency, computation, and power consumption. However, many applications require both authentication and

encryption; therefore the proposed protocol's design is based on the authentication and encryption security option. The authentication and encryption security option encrypts the data payload and authenticates the packet with a MAC. The MAC is computed over the packet header and the encrypted data.

The authentication and encryption security option utilizes a single, symmetric key that is shared among a collection of sensor network nodes. Before transmitting a packet, each node first encrypts the data and applies a strong unforgeable harsh (MAC) to protect the data integrity. The receiver verifies that the packet was not modified in transit by recomputing the MAC and then decrypts the message.

More specifically, the authentication and encryption security option uses a cipher block chaining (CBC) scheme called Skipjack [63] along with a specially formatted 8-byte initialization vector (IV) to encrypt its data. The structure of the IV is $Dest||\text{AM}||Len||Src||Ctr$, where *Dest* is the destination address, AM is the active message handler type, *Len* is the data length, *Src* is the source address, and *Ctr* is the counter value. The active message handle types are similar to port numbers in TCP/IP and specifies the appropriate handler function to extract and interpret the message on the receiver. *Ctr* is a 2-byte counter that starts at 0 and increments by 1 after each message sent by the sender. In addition to the CBC scheme used as an encryption mechanism, it is used to authenticate the entire packet. CBC-MAC [64], the authentication mechanism used for TinySec security protocol, allows programming code to be reused due to the similar structure of the encryption and authentication mechanism.

2.3 Power Estimation Strategies for Sensor Network

As the interest for research in WSNs increase, the demand for new tools to aid these researchers also increase. The majority of the developed simulation environments are geared towards providing varying degrees of scalability, realism and detail for

understanding the behavior of sensor networks. But, perhaps the most important simulation environment is the measurement of the power consumption. Surprisingly, to date, there are only a few power estimation models available.

In this section we review two power estimation strategies for sensor networks: *direct measurement* and *model simulation*, and briefly discuss some power estimation simulation models.

2.3.1 Direct Measurement

Direct measurement entails deploying sensors into a network and measuring the power consumption. Often this is considered a field experiment and the inputs of the sensor nodes are stimulated from the environment and radio communication from other sensor nodes. These inputs are real and not generated by a model during simulation or lab experiment. Simulated inputs are considered synthetic inputs.

Measurement of the power consumption is generally performed with an instrument (e.g., oscilloscope, digital multi-meter). Data collected by the sensors is either on-line or stored in a recorded log for post analysis. The trace or recorded log can consist of either measurable values (e.g., current, voltage, etc) or indirect measurements (e.g., number of packets, I/O activity, packet transmission time, etc).

With direct measurement, it is difficult to measure the power consumption when the number of nodes is large. In addition, this method is costly in terms of time and money, since it requires actual node deployment and node component measurement to estimate the power consumption. An easier and effective method to estimating the power consumption is software simulation.

2.3.2 Model Simulation

Software simulation is perhaps the best method to estimate the power consumption of a large sensor network. Simulation entails a synthetic model of the sensor node and network to predict sensor's behavior and lifespan. During the simulation, power related information is recorded. The nature of the power related information can

be very abstract (e.g., estimation of node duty cycle and communication rates) or detailed (e.g., estimation of low-level requirements of the CPU, radio, sensor and other peripherals). Nevertheless, the power relevant information is used to determine the power consumption of the nodes and network.

Generating inputs for simulations often requires synthetic environment models or captured traces of previous measurements of network traffic. Developing an accurate power model is critical for sensor networks because they are limited in power and generally operated by batteries. In addition, an accurate power model allows the developer to tune their applications before deployment in real environments. However, constructing synthetic environmental models is a challenging task which was apparent in the Great Duck Island sensor network deployment [65], where nodes significantly under-performed their expected lifetime.

Many existing simulation environments allow researchers to study various dynamics such as communication overhead, network behavior, and scalability; but few have focused on measuring power consumption. Conventional network simulators such as ns2 [66] focuses on the behavior of network protocols and fails to capture the operation of endpoint nodes in detail. Another drawback is that ns2 provides implementations of the 802.11 medium access control/physical layers, while many sensors networks employ nonstandard wireless protocols. Simulation environments such as SensorSim [67] and SENS [68] have considered power consumptions into their models. However, the two simulation environments incorporated simple power usage and battery models and does not appear to have been validated against actual hardware and real applications [69]. Power simulation tools such as EMSIM [70] and JouleTrack [71] were developed to measure the energy in embedded systems. EMSIM is designed for embedded systems with the StrongARM microprocessor and simulates the StrongARM instructions-set, memory, Universal Asynchronous Receiver/Transmitter (UART) and other peripherals connected to the processor. On the other hand, JouleTrack estimates only the microprocessor energy consumption. The EMSIM and JouleTrack tools perform accurately in energy profiling, but are specifically designed for simulating a single node's energy.

For this thesis, we are interested in estimating the power consumption for a large scale sensor network, thus we select the PowerTOSSIM power consumption simulator to estimate the energy consumption. PowerTOSSIM provides an accurate per node power estimation by profiling various hardware peripherals such as radio, EEPROM, LEDs and CPU. In addition, PowerTOSSIM is equipped with a graphical user interface to control and monitor the sensor environment. These attractive features lead to use of PowerTOSSIM to evaluate the energy consumption for the LLSP and TinySec security protocols.

2.4 Problem Formulation

In this section, we formulate the problem with implementing an efficient security protocol for WSNs. First, wireless channels are inherently unreliability due to its transmission medium. Secondly, wireless channels require at least equal, and often a higher level of security compared to its wired counterpart.

However, the main concern with WSNs is the fact that they are constrained in memory, processing and energy. Due to these extreme constraints, designing an efficient link layer security protocol is challenging. The proposed LLSP security protocol uses these inherent sensor network limitations to its advantage and provides an energy efficient link-layer security architecture for WSNs. Conventional security protocols tend to be conservative in their security guarantees and often require large communication overhead. However, LLSP is audacious and guarantees security services such as message authentication and access control, confidentiality, and replay protection while reducing the security communication overhead. To evaluate the performance of the LLSP, the energy consumption and throughput is compared with a prominent security protocol called TinySec. The energy consumption is estimated with the PowerTOSSIM simulator. Our experiential results comparison between the LLSP and TinySec security protocol demonstrate that LLSP is secure, efficient, and reduces the security overhead energy consumption by 15%.

2.5 Summary

In this chapter we first described the system models that complement our proposed security protocol. We also looked at prominent operating systems used to control these sensors. Then we reviewed power estimating strategies for sensor networks. Finally, we defined the problem and challenges of implementing an efficient security protocol for WSNs.

For this thesis, we assumed a hierarchical network consisting of cluster heads and regular sensor nodes. The wireless medium is error-free and not affected by propagation effects. Each node has four basic operations: *sense*, *transmit*, *receive*, and *listen*.

Operating systems such as SOS and MagnetOS are superior operating systems for sensor networks. However, TinyOS operating system is best suited for this thesis due to its built-in features. TinyOS provides a simulator, a security protocol and a graphical user interface which allow users to visually debug, test and analyze algorithms in a controlled environment. Most importantly, TinyOS features the PowerTOSSIM energy consumption simulator. PowerTOSSIM allows the user to measure an accurate power consumption of specific hardware peripherals such as radio, EEPROM, CPU and LEDs.

We also reviewed power estimation strategies for sensor networks. Direct measurement entails measuring the power consumption of each hardware peripheral manually. This approach is difficult and infeasible when considering a large scale network. An easier method is by simulation modeling. A model of the sensor node and network is used to stimulate the sensor nodes with synthetic inputs. Power related information is recorded and used to estimate the power consumption.

Finally, we define the problem with implementing a security protocol for WSNs. The major constraints are the limited processing, memory and energy of these nodes. However, the proposed LLSP security protocol overcomes these constraints and efficiently provides security services such as message authentication, confidentiality and replay protection.

In the next chapter we discuss the design and security services of the LLSP security protocol. In addition, we analyze the energy consumption and throughput of the LLSP and TinySec security protocols.

CHAPTER 3

LLSP Security Protocol and Evaluation

In an effort to provide an energy efficient security protocol, we propose the link-layer security protocol (LLSP). The main feature of the LLSP protocol is its ability to offer the same security services of the TinySec protocol plus provide replay protection, while reducing the energy consumption in the network. The reduction in the energy consumption is achieved by reducing the security overhead per packet by 2 bytes. In addition to reducing the energy consumption in the network and reducing the security overhead, the LLSP protocol is flexible, transparent and maximizes memory resources. The LLSP is not limited to any particular encryption scheme. However, it is strongly recommended that only well-known symmetric algorithms are applied. The LLSP protocol is also transparent to the other network layers and application users. The user can run secure applications without knowledge of the network configuration. Finally, the LLSP protocol maximizes the memory usage by reusing programming code. In this chapter, we discuss the design and performance of the LLSP protocol.

In Section 3.1, we review the security services provided by LLSP, which include message authentication, access control, confidentiality and replay protection. Section 3.2, reviews specific details regarding the LLSP design. Lastly, in Section 3.3, we thoroughly evaluate the LLSP security protocol with a security and performance analysis.

3.1 LLSP Security Services

WSNs are more vulnerable to security attacks because it is based on a broadcast medium. Consequently, an adversary can eavesdrop, intercept, inject and alter transmitted data if an appropriate transceiver is used. Furthermore, an adversary may use a radio transceiver and a powerful workstation to interact with the network from a distance. Such interactions may be used to perform a power consumption attack, which involve an adversary repeatedly transmitting packets to nodes within the network. Theses actions consume vital energy, waste the network bandwidth and shorten the longevity of sensor nodes in the network.

In order for WSNs to be widely deployed, an adequate security protocol must be integrated into the sensor network design. The proposed LLSP security protocol addresses the security issues mentioned in Section 1.1.3. In addition, the LLSP security protocol guarantees message authentication, access control, message confidentiality and replay protection. In the following subsections we discuss the security services of the LLSP security protocol.

3.1.1 Message Confidentiality

WSNs are based on a broadcast medium, therefore message confidentiality is an extremely important security service to include in the network design. Data encryption is a method of achieving message confidentiality when transmitting data through an unsecured medium. For the LLSP security protocol, we propose the Advance Encryption Standard (AES) [72] with cipher block chaining (CBC) mode of operation as the data encryption scheme. A depiction of the CBC encryption scheme is shown in Figure 3.1, where AES is the encryption scheme. As shown in the figure, the initial plaintext *P1* is bitwise exclusive-or (XOR) with an initialization vector (IV). The result is encrypted with a shared key K to produce the initial ciphertext *C1*. The second round of the encryption scheme is identical to the first round except the resulting ciphertext *C1* is used as the IV. This process is repeated for the desired number of rounds of the encryption method.

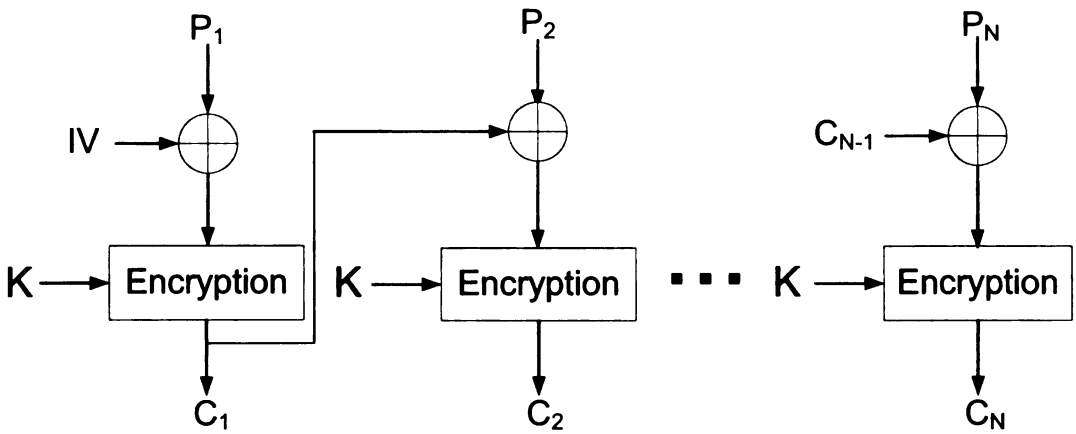


Figure 3.1. Cipher Block Chaining (CBC) mode encryption

The unique design of AES-CBC provides semantic security, which implies that encrypting the same plaintext twice, will produce two different ciphertexts. Semantic security is achieved by adding a unique IV to the encryption scheme. As shown in Figure 3.1, the IV is a side input to the encryption algorithm. The IVs are used to provide variation to the encryption process when there is little or no variation between the set of messages. In Section 3.2.1, we discuss the specific format of the IV used in LLSP security protocol design.

3.1.2 Message Authentication

Due to the unreliability and random characterization of wireless channels, they are more vulnerable to transmission errors than its wired counterpart. Typical communication protocols provide packet error checking with a cyclic redundancy check (CRC). A CRC is a type of hash function used to produce a checksum, which is a small, fixed number of bits appended to the message packet. To detect transmission errors, the sender computes a checksum over the packet. The receiver recomputes the checksum and verifies it with the received checksum field. If the two fields are equal, the receiver accepts the message and rejects it otherwise. However, a known flaw of CRC is that it does not protect against malicious modifications or forgery of packets.

To guarantee message authentication and access control, LLSP uses a message

authentication code (MAC). Message authentication prevents unauthorized nodes from participating in the network and ensures received messages are not altered, thus inherently assuring the message contains no errors. A MAC is essentially a cryptographically secure checksum of a message. Computation of the MAC is based on a cryptographic hash function and a secret shared key between the sender and receiver. Conventionally, the MAC for a message m can be represented as

$$MAC_m = H(K, m||Ctr), \tag{3.1}$$

where K is the shared key between the sender and receiver, Ctr is the counter value and H is the secure hash function, which is a one way function with variable length input and fixed length output.

Similar to CRC's computation structure, the MAC is also computed and appended to the message m before transmission. The receiver and sender share a secret key, therefore once the receiver receives the message, the receiver can recalculate the MAC. If the two MACs are equal, then the receiver keeps the packet and discards the packet if the MACs are not equal. The hash function ensures that, if an adversary alters a valid message or injects a malicious message, the receiver will reject the message because the receiver is unable to recompute the correct MAC value.

For LLSP security protocol, we propose the Cipher Block Chaining Message Authentication Code (CBC-MAC) to provide message authentication and access control. A depiction of the CBC-MAC encryption scheme is shown in Figure 3.2. As shown, the CBC-MAC uses a scheme similar to CBC, but the IV is initialized to zero. The similar computation method allows for code reuse thus reducing the memory usage.

3.1.3 Replay Protection

Due to the limited number of states that each node can maintain in its memory, replay protection is difficult to include in WSNs. In a replay attack, an adversary eavesdrops between two authorized sensor nodes and replay the message to the receiver at a later time. Typically, a counter value is used to maintain record of the received messages

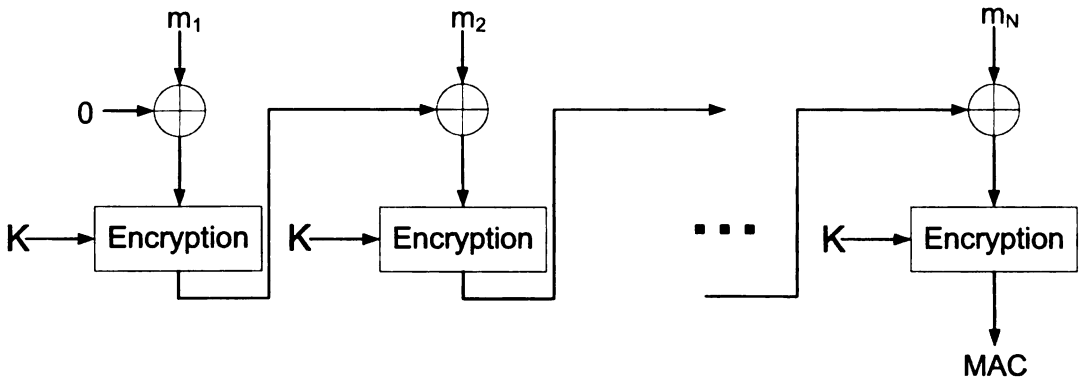


Figure 3.2. Cipher Block Chaining Message Authentication Code (CBC-MAC)

from a node. If the authorized receiver has a record of the received message, it can detect a replayed message and reject it. But, if there is no record of the received message then the receiver will accept the message again, consequently increasing the energy consumption. For large WSNs it is impractical for each node to maintain record of each senders message count. However, if the sensor node has knowledge of the network topology and power efficient routing, then counters are only necessary for the number of nodes directly in its communication range. Typically, the number of sensors in a communication range is small. Therefore, it is practical for a sensor node to maintain a counter for each node that is in its communication range.

The proposed LLSP security protocol maintains a synchronous 4-byte counter between the sender and receiver pair. The feedback shift register (FSR), shown in Figure 3.4, is used to update the 4-byte counter. Recall that the TinySec security protocol maintains a 2-byte counter by transmitting the counter value in each message packet. The FSR design allows energy to be saved by eliminating the transmission of the counter bytes in each message packet.

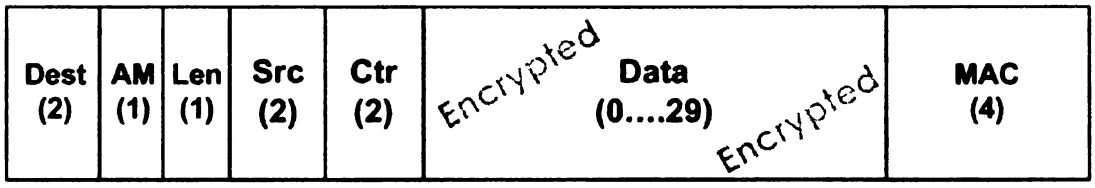
3.2 LLSP Design

The design of LLSP is based on the TinyOS's security protocol, TinySec. The two protocols are similar with important differences, namely the packet format and the

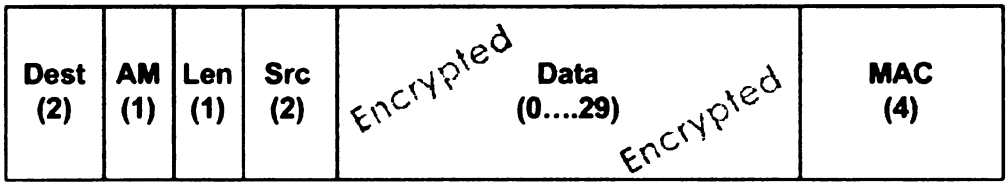
use of a feedback shift register as a counter. In this section, we introduce the packet format of the LLSP security protocol and explain the differences between the two packet formats. We also investigate the efficiency of the feedback shift register.

3.2.1 LLSP Packet Format

The packet format for the LLSP security protocol is based on the packet format of the TinySec protocol. Figure 3.3, depicts both security protocols. As shown, the common fields between the two protocols are the destination address (*Dest*), active message type (*AM*), the data length (*Len*) and the source address (*Src*). The two security protocols differ with the counter value (*Ctr*). As shown, the counter value is absent from the LLSP security protocol packet design. In Section 3.3, we explain that the removal of the counter bytes per message packet reduces the energy consumption by 15%.



(a) TinySec – Authentication & Encryption packet format



(b) Link Layer Security Protocol packet format

Figure 3.3. TinySec protocol and Link Layer Security Protocol packet formats. The byte size of each field is indicated below the label. In both packet formats, the grided area is encrypted.

Although the counter value is not included in the packet format for LLSP, it is included in the IV and needed for the computation of the MAC. The structure of the IV consists of *Dest*, *AM*, *Len*, *Src*, and *Ctr* bytes appended together. The counter value is included in the structure of the IV to add variation to the encryption, which will reduce the risk that the IV is repeated. IV reuse may severely compromise the security of the network. Similarly, the counter value is included in the computation of the MAC. Calculation of the MAC for the LLSP protocol is shown in Equation 3.2,

$$MAC = H(K, Dest||AM||Len||Src||Ctr||Data), \tag{3.2}$$

where *Data* is the encryption of the sensor reading or other information.

As shown in the packet format, the packet header (*Dest*, *AM*, *Len*, and *Src*) and the MAC are not encrypted. The benefits of not encrypting the packet header and MAC outweighs the benefits of keeping them a secret. Early rejection, a power saving mechanism for sensor nodes, where the nodes turns off its sensor radio after determining the message is not addressed to it. However, if the packet header is encrypted, early rejection cannot be invoked until the packet header is decrypted. If an adversary wants to disturb the sensor network, it can repeatedly transmit messages to the sensor thus forcing the sensor to consume energy by decrypting pointless messages. Early rejection can also be achieved if the MAC is not encrypted. This allows the receiver to determine the authenticity of the message with minimum energy consumption since the decision can be made without requiring the decryption of the data packet. The LLSP security protocol also allows the receiver to determine the number of lost packets based on the correct counter value that generates the MAC.

Furthermore, the LLSP security protocol reduces the energy consumption without decreasing the security, by reducing the security overhead by 2 bytes. As shown in Figure 3.3, the packet format for the LLSP security protocol does not include the counter value field. Each sender and receiver pair maintain a synchronous counter generated through a feedback shift register (FRS) shown in Figure 3.4. As a result,

the counter value is not transmitted with the message packet. In [73], the authors state that the transmission of 1 bit consumes about as much power as executing 8,000-1,000 instructions for the Mica2 mote. LLSP's 2-byte security overhead reduction is equivalent to not executing 12,800-16,000 instructions for each packet, which equates to less consumption of energy and an increase in network lifespan. In the following subsection we discuss the efficiency of FSR.

3.2.2 Linear Feedback Shift Register

The feedback shift register (FSR) has found use in a variety of applications ranging from a pseudo-random number generator to a fast digital counter. In short, a FSR is a sequential shift register with combinational logic that causes it to pseudo-randomly cycle through a sequence of binary values. Although FSRs are simple to design and implement, they are based on a complex mathematical theory [74]. FSRs can be implemented in software and hardware. Generally, the hardware implementation of FSRs are simple and fast but their software implementations are not as efficient. In [75], the authors introduce a efficient software implementation of a FSR.

Recall that the communication operation is the dominant energy consumer in a WSN. To save energy it is desirable to transmit and receive less communication overhead. The TinySec security protocol synchronizes the sender and receiver pair by transmitting the 2-byte counter with each message packet. In an effort to reduce the energy consumption per message packet, the LLSP security protocol maintains a synchronous 4-byte counter between each sender and receiver pair with a FSR, which is shown in Figure 3.4. Using the FSR instead of transmitting the 2-byte counter has its advantages. The energy cost of implementing the FSR is minimal compared to transmitting a 2-byte counter value per message packet. Also, the 2-byte counter increase of the LLSP security protocol design alleviates the IV reuse problem. A 2-byte counter guarantees an IV will not be reused until 2^4 packets are sent, whereas a 4-byte counter guarantees an IV will not be reused until 2^5 packets are sent. This increase doubles the amount of packets that can be sent before an IV is reused.

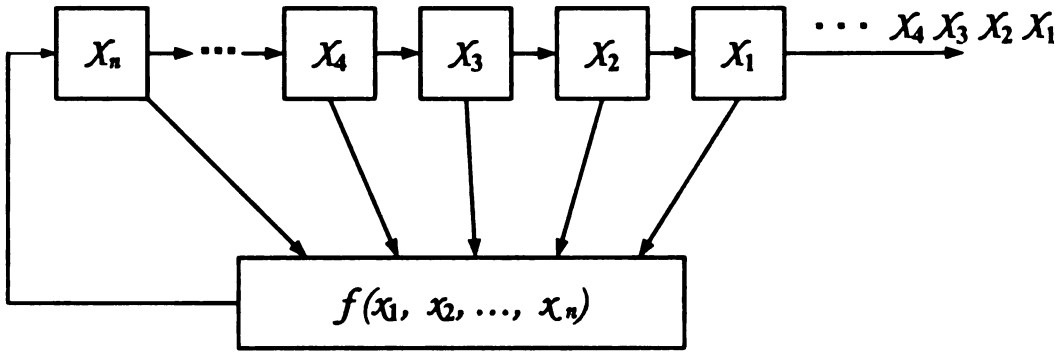


Figure 3.4. The structure of a feedback shift register

3.3 LLSP Evaluation

To evaluate the LLSP security protocol, we analyze the security and performance between TinySec and LLSP security protocols. In Section 3.3.1, we argue that LLSP is secure and efficient. In Section 3.3.2, we discuss the energy and throughput performance of both security protocols.

3.3.1 Security Analysis

The LLSP security protocol focuses on three data security services: message authentication, replay protection, and message confidentiality. There are many cryptographic algorithms that provide all three security services and LLSP is not limited to any cryptographic algorithm. It is strongly recommended that only well-known symmetric cryptographic algorithms be applied to ensure sensor network security and implementation efficiency.

The length of the IV has a dramatic affect on both the security and energy consumption of the security protocol. If the IV is too long there will be unnecessary bits added to the packets, which translates to significant cost in energy and bandwidth. At the same time, if the IV is too short there is a risk of IV reuse and then the security is compromised. The LLSP security protocol uses a 10-byte IV structure, where only 6 bytes are transmitted in each packet. The 4-byte counter value is not transmitted

and is used to add variation to the encryption process and reduce the risk of IV reuse.

The length of the MAC is directly related to the security of the MAC. Conventional security protocols uses MAC length sizes of 8, 10, 12, 16 and 20 bytes [20]. However, for a WSN, a 4-byte MAC is sufficient. A 4-byte MAC implies that an adversary has a 1 and 2^{32} chance of blind forging. In other words, an adversary needs to repeatedly send packets to an authorized receiver about 2^{31} times to achieve the correct MAC. Typically, sending 2^{31} packets to a receiver with higher bandwidth is trivial, but for a WSN, the application bandwidth is much smaller. As an example, the Mica2 sensor node that features a low-powered radio from Chipcon, can transmit at a data rate of 19.2kbps. At this slow speed, an adversary can only transfer 40 forgery attempts per second. Thus, it would take about 20 months for the adversary to transfer 2^{31} forgeries. Furthermore, the Mica2 sensor node operating at full power can only run for two weeks before it uses all of its battery resources; therefore exceeding the battery life by more than 40 times.

Similar to most security protocols, the LLSP security protocol increases the computational and energy cost for each packet transmission. There are two major contributions to these costs: the extra computation time and energy needed for cryptography, and the larger packet size due to the security overhead. Fast symmetric cryptosystems such as AES-CBC, ensure only a modest increase in process and RAM. As shown in Figure 3.3, to add authentication and encryption to a message packet only requires 10 bytes of security overhead. However, even for non-secure operations, the destination address, active message type and data length fields are necessary and cost 4 bytes of communication overhead. Furthermore, a checksum is generally performed on message packets to detect transmission errors. A simple checksum such as CRC-16, requires a minimum of 2 bytes of communication overhead. In order to do this, the source address should be specified in the message packet which contributes to at least another 1 byte; thus a total of 7 bytes of communication overhead is necessary for non-secure operations. The LLSP security protocol only requires an additional 3 bytes of overhead compared to typical non-secure operation. The 3-byte increase is very modest considering the additional services (message authentication, replay

protection, confidentiality, and error checking) that is provided with LLSP security protocol.

3.3.2 Performance Analysis

The performance of LLSP and TinySec security protocols are measured by the energy consumption and throughput of the two protocols. First we discuss how the simulation experiment is implemented. Secondly, we analyze the energy consumption of the two protocols. Finally, we discuss the performance of the throughput of an error-free channel and a lossy channel for both security protocols .

A. Implementation

The LLSP and TinySec security protocols were both implemented with the TinyOS operating system and written in the nesC programming language. To implement the LLSP security protocol, we modified the TinySec programming code and incorporated new programming code for the feedback shift register.

The energy consumption and throughput of the LLSP and TinySec security protocols were both measured with the PowerTOSSIM simulator environment. The focus of this thesis is on the performance of the security overhead, thus the energy consumption is based solely on the transmission of the security overhead (zero bytes of data payload) on an error-free radio channel.

The performance of the throughput is based on a data payload of 8 bytes per message and is simulated on two different channels: an error-free radio channel and a bit-error lossy radio channel. For this thesis, the throughput is defined as the measure of successful received packets per second per node in the network. The error-free radio channel measures the throughput in an ideal environment, where every bit transmitted is received without error. In contrast, the lossy model measures the throughput in a practical environment, where every bit transmitted is not received perfectly. The lossy radio model places each node in a directed graph with TOSSIM's LossyBuilder tool. Given a physical node topology, the LossyBuilder generates loss rates for each node

	Application Data (bytes)	Packet Overhead (bytes)	Total Size (bytes)	Time to Transmit (ms)	Latency Reduction
TinySec	24	44	68	28.3	-
LLSP	24	42	66	27.5	2.95%

Table 3.1. Table listing the expected latency reduction due to decrease in security overhead

pair by sampling Gaussian packet loss probability distributions [76]. The packet error rates are then translated into independent bit error rates. The lossy radio channel captures interference and corruption, but is not capable of modeling noise.

B. Energy costs

To analytically estimate the cost of the cryptographic service, we first calculate the effect of packet lengths between the LLSP and TinySec security protocols. Recall that TinySec security overhead is 2 bytes larger than the LLSP security overhead. Longer security overhead affect the sensor network in several ways: first, it reduces bandwidth; second, it increases latency because of the fairly slow communication channels; third, it increases the energy consumption because the communication radio must be turned on for a longer period when transmitting longer overhead. We first calculate and compare the expected latency due to the LLSP and TinySec security overheads. Table 3.1 shows the extra time needed to transmit a packet using TinySec. We expect the LLSP to reduce the latency by approximately 3%. Note that sending a packet involves more than just sending the associated header and data payload. Media access control information such as start symbol and synchronization bytes are also transmitted. This will cause a discrepancy between the theoretical and simulated energy overhead cost for LLSP security protocol.

To determine the theoretical energy overhead cost for the LLSP and TinySec security protocols, the power is calculated using Equation 3.3

	Theoretical (mJ)	Simulated (mJ)
TinySec Protocol	0.322	0.5283
Link Layer Protocol	0.269	0.4510
Improvement	16.67%	14.63%

Table 3.2. Energy consumption results of transmitting security overhead.

$$P = I * V, \tag{3.3}$$

where P is the power, I is the transmission current and V is the voltage. For all cases, the transmission current used is 21.48mA and the voltage is 3v. The energy is computed by multiplying the power found in Equation 3.3, by the packet transmission time. The packet transmission time depends on the data rate of the Mica2 sensor node platform, which has a data rate of 19.2 kbps. Table 3.2 summaries the theoretical results and shows that the LLSP security protocol reduces the energy cost of the security overhead by approximately 17%. This result is expected since the reduction in bytes of the security is approximately 17% as well.

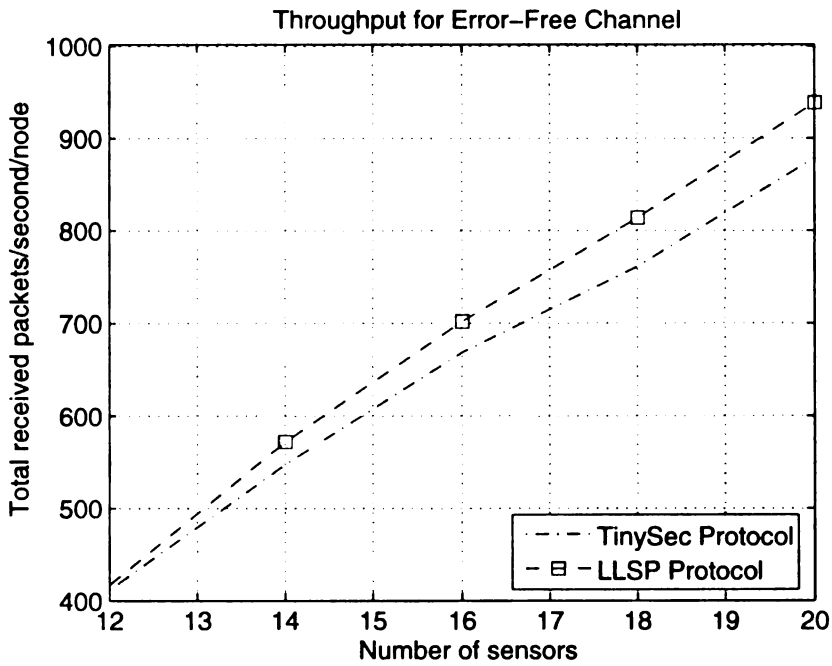
The simulation results in Table 3.2 are based on the PowerTOSSIM simulator environment. The measurement of the energy consumption for each protocol is based on the transmission of a message packet consisting of only the security overhead. The results shown in Table 3.2, state only a 15% improvement between the two protocols. The discrepancies between the theoretical and simulated results are due to the hidden additional bytes of data transmitted during the simulation calculations. During simulation, the sender node always transmits start symbol bytes and pulse strength bytes. These additional bytes cause for only a 15% reduction in energy cost for the security overhead of the LLSP security protocol.

C. Throughput

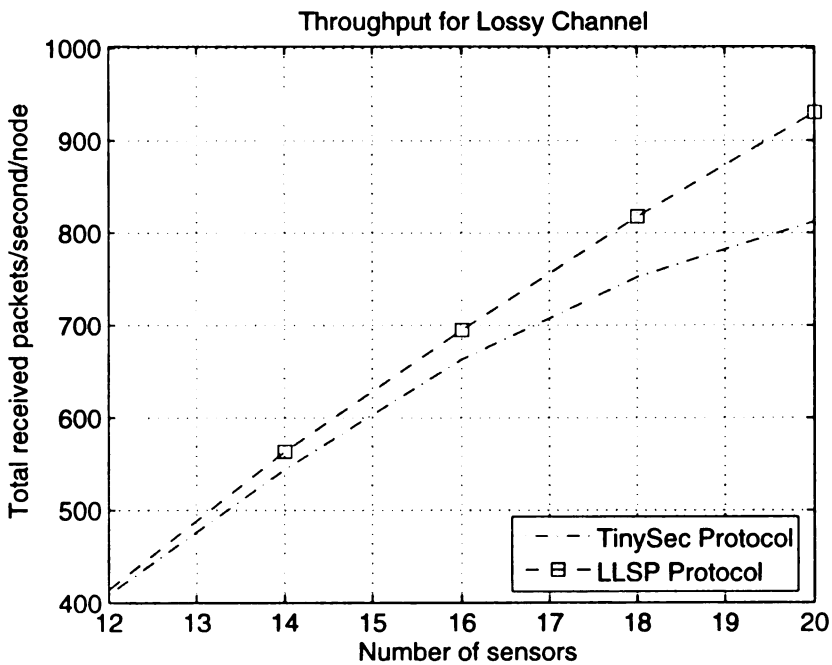
To measure the maximum throughput of the TinySec and LLSP security protocols, the total number of successful received packet in the network were calculated for a

30 second time period. In this experiment, the network was configured such that each sensor node in the network simultaneously transmit packets. Since the number of senders affects the channel utilization, the number of senors in the network is varied. This allows a complete characterization of the throughput at different regimes. The data payload for each security protocol is 8 bytes. As stated in Section 3.3.2, the throughput of both protocols are measured on an error-free channel and a lossy channel. The results of the error-free channel and lossy channel are in Figure 3.5.

First we investigate the throughput performance of the error-free channel. As shown in Figure 3.5(a), the throughput of the LLSP security protocol gradually begins to outperform the TinySec security protocol as the number of sensors in the network increase. With 20 sensors in the network, the LLSP security protocol increased the throughput by approximately 6%. Recall that in an error-free channel both protocols receive all packets without error. As a result, a small number of packets are required to be retransmitted due to the CSMA/CA protocol. However, the performance of the throughput of a lossy channel shown in Figure 3.5(b), illustrates a greater improvement in throughput. As the number of sensors increases, the throughput of the LLSP security protocol outperforms the TinySec security protocol by 13%. The increase in the throughput is due to the fact that a lossy channel models bit errors, therefore more packets require retransmission. Overall, reducing the security communication overhead increases the throughput for both simulated channels, but for a lossy channel the performance of the throughput is larger.



(a) Throughput for Error-Free Channel.



(b) Throughput for Lossy Channel.

Figure 3.5. Throughput, plotted as a function of the number of sensors.

CHAPTER 4

Conclusion and Future Work

4.1 Conclusion

In this thesis we analyzed the importance of a security service in WSNs, discussed the major challenges of incorporating a security feature and proposed an energy efficient link-layer security protocol that upholds the sensor node security requirements while reducing the energy consumption.

WSNs has a great potential to revolutionize health care, homeland security and environmental monitoring from how we know it today. However, the major challenges of incorporating a security feature can restrict wide deployment of applications that requires a high degree of data communication integrity. The severely constrained memory and energy resources of the WSN infrastructure prohibits traditional security techniques from being directly applied to the network. Furthermore, WSNs are more vulnerable to security attacks than its wired counterpart due to its broadcast nature. The lack of physical boundary makes eavesdropping, interception, injection and alteration of the transmitted data more attainable. For WSNs it is desirable for the security protocol to be simple but effective in design.

The proposed LLSP protocol is an energy-efficient and secure link-layer protocol for WSNs. The design of LLSP is based on the TinySec security protocol, but reduces the energy consumption per message packet by 15%. This is achieved by reducing the security communication overhead by 2 bytes. LLSP security protocol disregards the 2-byte counter values in the security overhead, but maintains a synchronous counter for each sender and receiver pair with a feedback shift register. In addition to the reduction in energy consumption, this design also increases the throughput for

both the error-free channel and the lossy channel. The error-free channel showed an increase in throughput by 6% and the lossy channel showed an increase in throughput by 13%. Overall, the LLSP security protocol is a simple, but secure protocol that can be integrated into the existing applications with a minimal application overhead.

4.2 Future Work

This thesis presented simulation results of the energy consumption and throughput. However, in the future an investigation of the latency of the LLSP security protocol should be carried out. Measurement of the latency is important because many applications have a multi-hop topology. In addition to performing the simulation of the latency, a key mechanism in the link-layer security design is worthy of a thorough investigation. Keying mechanisms handle the distribution and sharing of cryptographic keys throughout the network; therefore, an energy efficient key management scheme would complement and strengthen our energy efficient link-layer security protocol.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 126–137, 2003.

[2] M. Kuoilehto, M. Hannikainen, and T.D. Hamalainen, "A survey of application distribution in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, pp. 774–788, Oct 2005.

[3] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE Transactions on Mobile Computing*, 2002.

[4] A. Bharathidasas and V. Anand, "Sensor networks: An overview," *Technical report, Dept. of Computer Science, University of California at Davis*, 2002.

[5] Chee-Yee Chong and S.P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of IEEE*, pp. 1247–1256, Aug 2003.

[6] J. Hill, "System architecture for wireless sensor network," *PhD Dissertation, University of California, Berkeley*, 2003.

[7] I. Khemapech, I. Duncan, and A. Miller, "A survey of wireless sensor networks technology," *Proceedings of the 6th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, June 2005.

[8] J. Feng, F. Koushanfar, and M. Potkonjak, "System-architecture for sensor networks issues, alternatives and directions," *ICCD'02*, 2002.

[9] J.N. Al-Karaki and A.E. Kamal, "Routing techniques in wireless sensor networks: A survey," *IEEE Wireless Commuincations*, pp. 6–28, Dec 2004.

[10] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," *Proceedings of the 5th ACM/IEEE Mobicom*, pp. 174–185, August 1999.

[11] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Negotiation-based protocols for dissemination information in wireless sensor networks," *Wireless Networks*, pp. 169–185, 2002.

[12] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Proceeding of ACM MobiCom*, pp. 56–67, 2000.

[13] H. Wang, Z. Cen, T. Li, and M. Mutka, "A topology-aware routing protocol for an ad-hoc network with multiple sinks," *Electro/Information Technology Conference (EIT)*, 2006.

[14] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *Proceeding of 33rd Hawaii International Conference on System Science*, January 2000.

[15] S. Lindsey and C. Raghavendra, "Pegasis: Power-efficient gathering in sensor information systems," *IEEE Aerospace Conference*, pp. 1125–1130, 2002.

[16] A. Manjeshwar and D. Agarwal, "Teen: A routing protocol for enhanced efficiency in wireless sensor networks," *15th International Parallel and Distributed Processing Symposium*, pp. 2009–2015, April 2001.

[17] Y. Xi, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad-hoc routing," *Proceeding of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, July 2001.

[18] B. Chen, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless Networks*, pp. 481–494, 2002.

[19] J. Ren, T. Li, and D. Aslam, "A power efficient link layer protocol (LLSP) for wireless sensor network," *Military Communication Conference*, pp. 1002–1007, Oct 2005.

[20] R. Venugopalan, P. Ganesan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiui, "Encryption overhead in embedded systems and sensor network nodes: Modeling and analysis," *2003 International Conference on Compliers, Architectures and Synthesis for Embedded Systems*, pp. 188–197, 2003.

[21] B. Preneel and P. van Oorschot, "On the security of iterated message authentication codes," *IEEE Transactions on Information Theory*, pp. 188–199, January 1999.

[22] J.P. Walters, Z. Liang, W. Shi, and V. Chaudhary, *Security in Distributed, Grid, and Pervasive Computing*, chapter Wireless Sensor Networks Security: A Survey, CRC Press, 2006.

[23] A. Perrig, R. Szewczyk, Wen V, D. Culler, and J.D. Tygar, "SPINS: Security protocols for sensors networks," *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks*, July 2001.

[24] Crossbow Mica2 Datasheet, *http : //www.xbow.com.*

[25] Chipcon CC1000 Datasheet, *http : //www.chipcon.com.*

[26] T. Park and K.G. Shin, "Soft tamper-proofing via program integrity verification in wireless sensor networks," *IEEE Transactions on Mobile Computing*, pp. 297–309, June 2005.

[27] J.A. Stankovic, "Real-time communication and coordination in embedded sensor networks," *Proceeding of the IEEE*, pp. 1002–1022, July 2003.

[28] R.L. Rivest, "The `rc5` encryption algorithm," *Proc. 1st Workshop on Fast Software Encryption*, pp. 86–96, 1995.

[29] K. Zheng and Y. Pan X. Luo, and and Z. Wu, "Encryption algorithms comparisons for wireless networked sensors," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1142–1146, Oct 2004.

[30] D. Wheeler and R. Needham, "TEA, a tiny encryption algorithm," *http://www.ftp.cl.cam.ac.uk/ftp/papers/djw-rmn/djw-rmn-tea.html*, November 1994.

[31] G. Yuval, "Reinventing the travois: Encryption/MAC in 30 ROM bytes," *Proceeding of 4th Workshop on Fast Software Encryption*, 1997.

[32] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile commmunications: The insecurity of 802.11," *The Seventh Annual International Conference on Mobile Computing and Networking*, 2001.

[33] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesss in the key scheduling algorithm of rc4," *Lecture Notes in Computer Science*, 2001.

[34] Stubblefield A, J. Ioannidis, and A.D. Rubin, "Using the fluher, mantin and shamir attack to break wep," *Network and Distributed Systems Security Symposium (NDSS)*, 2002.

[35] "Ieee std 802.11i/d3.0," November 2002 (Draft Supplement to ISO/IEC 8802-11/1999(I) ANSI/IEEE Std 802.11, 1999 edition).

[36] IEEE standard 802.11i, "Draft supplement to standard for telecommunications and information exchange between systems `lan/man` specfific requirements-part 11: Wireless medium access control (mac) and physical layer specifications: Specifications for enhanced security," November 2002.

[37] C. Karlof, N. Sastry, and D. Wagner, "A link layer security architecture for wireless sensor networks," *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pp. 162–175, 2004.

[38] TinyOS, *http : //www.tinyos.net.*

[39] "Cellular digital packet data system specification, release 1.1," *CDPD Forum*, Jan 1995.

[40] GSM 02.09, "Digital cellular telecommunications systems (phase 2+): Security aspects," *European Telecommunication Standard GSM 02.09.*

[41] C. Gehrmann, T. Xydis, K. Nyberg, G. Muncaster, B. Austin, T. Baker, S. Blake-Wilson, D. Wiley, S. Wetzel, J. Larsson, B. Thomsen, and G. Seuron, "Bluetooth security white paper," *Bluetoth SIG Security Expert Group (http://www.bluetooth.com)*, April 2002.

[42] Y. Frankel, A. Herzberg, P.A. Karger, H. Krawezyk, C.A. Kunzinger, and M. Yung, "Security issues in a CDPD wireless network," *IEEE Personal Communications*, pp. 16–27, August 1995.

[43] A. Mehrotra and L.S. Golding, "Mobility and security management in the GSM system and some proposed future improvements," *Proceeding of the IEEE*, pp. 1480–1497, July 1998.

[44] M. Jakobson and S. Wetzel, "Security weaknesses in bluetooth," *In CT-RSA 2001*, pp. 176–191.

[45] B. Patel and J. Crowcroft, "Ticket based services access for the mobile user," *Third annual ACM/IEEE international conference on Mobile and computing and networking*, pp. 223–233, Sept. 1997.

[46] S. Czerwinski, B.Y. Zhao, T.D. Hodes, A.D. Joseph, and R.H. Katz, "An architecture for a secure service discovery service," *Fifth annual ACM/IEEE international conference on Mobile and computing and networking*, pp. 24–35, August 1999.

[47] L. Zhou and Z.J. Hass, "Secure ad hoc networks," *IEEE network, special issue on network security*, pp. 24–30, Nov. - Dec. 1999.

[48] T. Ylonen, "SSH - secure login connections over the internet," *Proceeding of the Sixth USENIX Security Symposium*, 1995.

[49] OpenSSL, *http : //www.openssl.org.*

[50] S. Kent and R. Atkinson, "Security architecture for the internet protocol," *RFC 2401*, November 1998.

[51] O. Younis, S. Fahmy, and P. Santi, "Robust communication for sensor networks in hostile environments," *Twelfth IEEE Internation Workshop on Quality of Service*, pp. 10–19, June 2004.

[52] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient apporach," *Proceeding of IEEE INFOCOM*, March 2004.

[53] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communication*, October 2002.

[54] S. Bandyopadhyay and E. Coyle, "An energy-efficient hierarchical clustering algorithm for wireless sensor networks," *Proceedings of IEEE INFOCOM*, April 2003.

[55] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coodination in sensor networks," *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking*, August 1999.

[56] H. Chan and A. Perrig, "Ace: An emergent algorithm for highly uniform cluster formation," *Proceedings of First European Workshop on Sensor Networks*, January 2004.

[57] H. Liu, T. Roeder, K. Walsh, R. Barr, and E.G. Sirer, "Design and implementation of a single system image operating system for ad hoc networks," *International Conference on Mobile Systems, Applications, and Services (Mobisys)*, June 2005.

[58] C. Han, and E. Kohler R. Rengaswamy, R. Shea, and M. Srivastava, "Sos: A dynamic operating system for sensor networks," *Proceedings of IPSN*, April 2005.

[59] D. Lymberopoulos and A. Savvides, "Xyz: A motion-enabled, power aware sensor node platform for distributed sensor network applications," *Proceedings of the Third International Conference on Mobile Systems, Applications, And Services (Mobisys)*, 2005.

[60] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, *Ambient Intelligence*, chapter `TinyOS`: An Operating System for Sensor Networks, Springer, 2005.

[61] D. Gay, P. Levis, R. Behren, M. Welsh, E. Brewer, and D. Culler, "The nesc language: A holistic approach to network embedded systems," *Programming Language Design and Implementation (PLDI)*, 2003.

[62] V. Scnayder, M. Hempstead, B. Chen, G. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pp. 188–200, 2004.

[63] E.F. Brickell, D.E. Denning, S.T. Kent, D.P. Mahler, and W. Tuchman, "SKIPJACK Review," *Interim Report*, July 1998.

[64] M. Bellare, J. Kilian, and P. Rogaway, "The security of the cipher block chaining message authentication code," *Journal of Computer and System Sciences*, pp. 362–399, December 2000.

[65] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," *Proceeding of the First European Workshop on Wireless Sensor Networks*, January 2004.

[66] K. Fall and K. Varadhan, *The ns manual*, *http : //www.isi.edu/nsnam/ns/doc/index.html*.

[67] S. Park, A. Savvides, and M.B. Srivastava, "SensorSim: A simulation framework for sensor networks," *Proceeding of MSWIM*, August 2000.

[68] S. Sundresh, W.Y. Kim, and G. Agha, "SENS: A sensor, environment and network simulator," *Proceeding of 37th Annual Simulation Symposium*, 2004.

[69] S. Park, A. Savvides, and M. B. Srivastava, "Simulating networks of wireless sensors," *Proceeding of the 2001 Winter Simulation Conference*, December 2001.

[70] T.K. Tan, A. Raghunathan, and N.K. Jha, "EMSIM: An energy simulation framework for an embedded operation system," *Proceeding of the International Conference on Circuits and Systems*, 2002.

[71] A.C. Amit Sinha, "`jouletrack` - a web based tool for software energy profiling," *Proceeding of the 38th Design Automation Conference*, 2001.

[72] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 2003.

[73] J. Hill, R. Szewczyk, S. Hollar, A. Woo, D. Culler, and K. Pister, "System architecture directions for networked sensors," *Proceedings of ACM ASPLOS IX*, 2000.

[74] R. David, *Random Testing of Digital Circuits: Theory and Application*, 1998.

[75] B. Tsaban and U. Vishne, "Efficient linear feedback shift registers with maximal period," *Finite Fields and their Applications*, pp. 256–267, 2002.

[76] P. Levis and N. Lee, *TOSSIM: A Simulator for TinyOS Network*, *http* : *//www.cs.berkeley.edu/~pal/pubs/nido.pdf*.