ENERGY EFFICIENT SECURITY FOR WIRELESS SENSOR NETWORKS

by

Abidalrahman Moh'd

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
June 2013

DALHOUSIE UNIVERSITY

DEPARTMENT OF ENGINEERING MATHEMATICS AND INTERNETWORKING

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “Energy Efficient Security for Wireless Sensor Networks” by Abidalrahman Moh’d in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Dated: 18 June 2013

External Examiner:	_________________________
Research Co-Supervisors:	_________________________
	_________________________
Examining Committee:	_________________________
	_________________________

Departmental Representative: _________________________

DALHOUSIE UNIVERSITY

DATE: 18 June 2013

AUTHOR: Abidalrahman Moh'd

TITLE: Energy Efficient Security for Wireless Sensor Networks

DEPARTMENT OR SCHOOL: Department of Engineering Mathematics and Internetworking

DEGREE: PhD CONVOCATION: October YEAR: 2013

____

Signature of Author

DEDICATION PAGE

I dedicate this thesis to my parents, beloved wife, and unborn baby.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis presents two main achievements. The first is a novel link-layer encryption protocol for wireless sensor networks. The protocol design aims to reduce energy consumption by reducing security-related communication overhead. This is done by merging security-related data of consecutive packets. The merging is based on simple mathematical operations. It helps to reduce energy consumption by eliminating the requirement to transmit security-related fields in the packet. The protocol is named the Compact Security Protocol and is referred to as C-Sec. In addition to energy savings, the C-Sec protocol also includes a unique security feature of hiding the packet header information. This feature makes it more difficult to trace the flow of wireless communication, and helps to minimize the effect of replay attacks. The C-Sec protocol is rigorously tested and compared with well-known related protocols. Performance evaluations demonstrate that C-Sec protocol outperforms other protocols in terms of energy savings. The protocol is evaluated with respect to other performance metrics including queuing delay and error probability.

The C-Sec operation requires fast encryption, which leads to a second major contribution: The SN-Sec, a 32-bit RISC secure wireless sensor platform with hardware cryptographic primitives. The security vulnerabilities in current WSNs platforms are scrutinized and the main approaches to implementing their cryptographic primitives are compared in terms of security, time, and energy efficiency. The SN-Sec secures these vulnerabilities and provides more time and energy efficiency. The choice of cryptographic primitives for SN-Sec is based on their compatibility with the constrained nature of WSNs and their security. The AES implementation has the best data-path and S-Box design in the literature. All SHA family members are implemented and compared to choose the most compatible with WSN constraints. An efficient elliptic-curve processor design is proposed. It has the least mathematical operations compared to elliptic-curve processors proposed for WSNs in the literature. It also exploits parallelism among mathematical operations to compute elliptic-curve point multiplication with minimal amount of clock cycles. SN-Sec is implemented using VHDL. Experimental results using synthesis for Spartan-6 low-power FPGA shows that the proposed design has very reasonable computational time and energy consumption.

LIST OF ABBREVIATIONS USED

AES	Advanced Encryption Standard
AM	Active Message
APIs	Application Programming Interfaces
ARQ	Automatic Repeat Request
BF	Bloom Filter
BGA	Ball Grid Array
BGCD	Binary Greatest Common Devisor Algorithm
C-Sec	Compact Security protocol
CBC	Cipher Block Chaining
CCM	Counter with Cipher block chaining Message authentication code
CFB	Cipher Feed Back
CPLD	Complex Programmable Logic Device
CRC	Cyclic Redundancy Check
CTS	Clear To Send
Ctr	Counter
Dest	Destination Address
ECC	Elliptic-Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman key exchange algorithm
ECDSA	Elliptic Curve Digital Signature algorithm
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GCM	Galois Counter Mode
Grp	Group
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ISA	Instruction Set Architecture
Len	Length
LLSP	Link Layer Security Protocol

M	Mode bit
MAC	Message Authentication Code
MD5	Message Digest 5
MH	Masked Header
MSB	Most Significant Bit
NIST	National Institute of Standards and Technology
OCB	Offset Code Book
OS	Operating System
PAN	Personal Area Network
PDoS	Path-based Denial of Service
PCB	Printed circuit Board
QoS	Quality of Service
Ran	Random number
RC5	Rivest Cipher 5
RISC	Reduced Instruction Set Computing
RTS	Request To Send
SEA	Scalable Encryption Algorithm
SEGC	Securities Exchanges Guarantee Corporation
SCID	Security Composition Identifier
SHA	Secure Hashing Algorithm
SN-Sec	Sensor Node Secure platform
SNEP	Secure Network Encryption Protocol
Src	Source Address
SoC	System on Chip
T-MAC	Timeout Medium Access Control
VHDL	Very high speed integrated circuit Hardware Description Language
WSN	Wireless Sensor Networks
ZPU	Zylin Processor Unit

ACKNOWLEDGEMENTS

I would like to thank my supervisors Dr. William Phillips, and Dr. Nauman Aslam, for their guidance, support, encouragement, and the countless hours they spent with me throughout the research of this work and publications. Dr. Phillips is a very friendly, helpful, understanding, wise, and worm-hearted person. Dr. Aslam is a very energetic, enthusiastic and active person. I am happy to be his first PhD student.

I would also like to thank the readers Dr. Bill Robertson and. Dr. Srini Sampalli. I am honored to have them in my committee and grateful for their guidance and encouragement throughout the process. I would also like to thank Dr. Anjali Agarwal for agreeing to be a part of the examining committee as the external examiner.

I owe special thanks to Dr. Ashraf AbuSharekh for many useful insights and help with the simulation experiments. I would also like to thank my PhD candidate friends Shadi Shehadeh and Omar Al-Qattami for their great assistance, as we spent many late nights together organizing the thoughts in my head.

Special thanks go to my family, especially my brothers and my sister, whom I have missed these many years, being away from home. I am also indebted to my aunt and her family for being supportive throughout the whole period of my PhD. I am very thankful for their prayers and continued support.

I would especially like to thank the person who has endured the hardship of my absence, and put up with my constant urgency to complete my work. I express heartfelt, unending gratitude to my wife and the mother of unborn child for being a pillar of support standing firm beside me throughout this journey.

Finally, I would like to express my sincere love and gratitude to my parents. I owe them more than anyone could possibly imagine, notwithstanding my academic achievements. I thank them for their love and infinite care, which has made the person I am today.

CHAPTER 1 INTRODUCTION

1.1 Motivation and Background

The future of wireless sensor networks (WSNs) is promising. They are extensively being deployed in many real-world applications within many contexts, like ubiquitous computing, ambient intelligence, and the Internet of Things (IoT) [1]. WSNs provide low cost solutions to a wide range of real-life challenges. It is expected that new WSN applications will have a greater share of the embedded systems market, with a growth rate of up to 50% per year [2].

Security is an important requirement for many WSN applications, like healthcare [3], structural [4], industrial [5] monitoring, military [6], and smart homes [7] . However, it is difficult to provide security in most of these applications due to many factors. Some of these factors are related to the constrained nature of WSNs, and others are related to environments where sensor nodes are being deployed. However, the most critical factor is energy. WSNs are expected to operate for long periods that extend to years using their limited-power battery. Batteries are difficult or infeasible to replace in most cases. This makes energy the most crucial resource in WSNs.

Wireless communication is a major source of energy consumption in WSNs. Applying security protocols has additional overhead on wireless communication. This overhead is classified in to two types. The first type is packets specifically generated for security protocol operations i.e. packets carrying security information or security control packets. This overhead occurs at the initial security setup stage, and it is repeated after periods to refresh keys or rerun authentication algorithms because of changes in the network. This type of overhead is considered small compared to the second type: the security data fields added to the headers of all communicated data packets. Many research efforts have been reported in the literature to improve the energy communication efficiency related to the security of wireless sensor networks, covering almost all aspects of security algorithms that involve communication overhead [8-16].

Another major source of energy consumption related to security is computational power overhead. Most of this power is spent on intensive mathematical computations of cryptographic primitives. Some of these primitives, such as symmetric encryption and hashing primitives, are frequently used to secure each data packet. Asymmetric primitives used to exchange keys and for digital signatures, which happens less often. Reducing the computational power consumption of these primitives can be achieved by implementing them in hardware rather than software. Hardware implementations are more time-efficient than software implementation as well.

The manufacturing cost is a major factor that makes it difficult to provide security for WSNs. Cost limit leads to constraints on the hardware platform. WSNs usually have simple and cheap controllers with limited processing power and memory size. Running computationally expensive primitives on them takes long time and consumes a lot of energy. Adding new hardware to implement those primitives will solve the energy and time overhead problems, but it increases the manufacturing cost.

Many other factors make it more difficult to provide adequate security for WSNs. WSNs are usually deployed in hostile environments, operate in an open wireless medium, and they are left unattended for their lifetime. In such conditions, sensor nodes can be easily accessed by an adversary without being discovered. This makes WSNs more vulnerable to security threats than traditional networks.

The research in this thesis is focused on two fundamental challenges: energy conservation and providing adequate security services for WSNs. In this regard, the goal is to design a secure encryption protocol based on a secure hardware platform to achieve the objectives of providing security and preserving energy. The problem of energy conservation is tackled at the physical hardware platform and data-link layer. This is done by designing the Compact Security protocol (C-Sec) at the link layer level, which relies on the Sensor Node Secure platform (SN-Sec) designed at the physical layer level.

The C-Sec [17] is a dual mode encryption protocol for WSNs, which is designed with an objective of reducing security-related communication with a slight increase in processing. Following Moore's law, dramatic reduction in energy consumption can be achieved from implementing the same hardware with a smaller process technology. In contrast, improvements to energy consumption of communication in the wireless medium are relatively small due to the modest improvements in transceiver technologies and the need to achieve acceptable levels of signal-to-noise ratio at specific distance. The trade-off between the energy costs of communication versus processing formulates the design philosophy of the C-Sec protocol. This protocol is based on reducing an energy-costly communication with a small increase in processing that has a much lower- and relatively decreasing energy cost. A detailed description of the C-Sec protocol design is presented in Chapter 4.

The C-Sec operation requires fast encryption primitives, which leads to a second major contribution. The SN-Sec is a 32-bit Reduced Instruction Set Computing (RISC) secure wireless sensor platform with hardware cryptographic primitives. The choice of cryptographic primitives for SN-Sec is based on their compatibility with the constrained nature of WSNs and their security. The Advanced Encryption Standard (AES) [18] implementation has the best data-path and S-Box design in the literature. All Secure Hashing Algorithm (SHA) family members are implemented and compared to choose the most compatible with WSN constraints. An efficient elliptic curve processor design is proposed. It has the least number of mathematical operations compared to elliptic curve processors proposed for WSNs in the literature. It also exploits parallelism among mathematical operations to compute elliptic curve point multiplication with minimal amount of clock cycles. SN-Sec is implemented using Very high speed integrated circuit Hardware Description Language (VHDL). Experimental results using synthesis for Spartan-6 low- power Field Programmable Gate Array (FPGA) shows that the proposed design has a very reasonable computational time and energy consumption.

1.2 PROBLEM STATEMENT

Implementation of security for WSNs is a very active research area with many challenges, especially the limitation of energy. Unfortunately, existing link-layer encryption protocols are unable to achieve an adequate level of security without a significant increase of energy consumption for computation and communication. Many of these protocols try to minimize communication costs by minimizing the size of the security related traffic, especially the fields added to the header of each packet transmitted in the wireless medium [8-16]. Some of these protocols proposed optional security levels that provide partial security services to minimize the overhead of security [10-12]. However, providing partial security leaves many vulnerabilities from which a WSN can easily be attacked.

This thesis presents a solution to the problem of link-layer encryption protocols communication overhead. This is done by proposing a link-layer encryption protocol that not only minimizes the communication overhead to zero, but also reduces the size of communicated packets to lower than the size of plain packets that do not implement encryption protocols at all. The significant reduction of communication overhead does not affect the security services provided by the proposed encryption protocol, but adds new security features that did not exist in any of the related protocols.

The work in this thesis addresses the problem of infeasibility of asymmetric encryption primitives. Security protocols avoid the use of asymmetric encryption algorithms to minimize computational power and latency, and replace them with symmetric lightweight encryption primitives that take less CPU cycles to run, consume less energy, and require less memory resources. Avoiding asymmetric encryption raises many security consequences related to key exchange, confidentiality, and authentication.

Many of the current security protocols rely on symmetric encryption primitives implemented in software, which run on WSN controllers with weak processing capabilities. Software implementations are costly in terms of computational time and energy, and rely on the security of the operating system. Some security protocols used hardware implementation of these primitives from “off the shelf” components used for environments less constrained than WSNs [19-21]. Secure design for unique WSN environments was not considered when they were manufactured. This makes sensor nodes vulnerable to node compromise and other various types of attacks. Building security protocols on top of insecure platforms will degrade their security, especially for WSNs, because they operate unattended, in hostile environments, and with unlimited access by adversaries. The work in this thesis addresses the security, energy-and time efficiency of sensor node platforms.

1.3 LIST OF CONTRIBUTIONS

The research presented in this thesis has a number of contributions related to two main topics that address energy saving, as well as WSN security. The first one is C-Sec, an energy efficient link-layer encryption protocol. The second is SN-Sec, a secure sensor-node platform design that provides secure key storage and energy efficient hardware based encryption primitives for C-Sec. These contributions are summarized here and discussed in detail in the following chapters:

- The design of C-Sec [17]: a link-layer encryption protocol that manages to provide all required security services of data authentication, integrity, confidentiality, semantic security, and replay protection. C-Sec provides all of these services with saving on energy consumption even over the plain packets without security, instead of increasing energy consumption like other protocols. A simulation model is provided to evaluate C-Sec and compare it with other related protocols using Castalia simulator [22], with evaluation metrics including communication energy consumption, error probability, and additional queuing delay analysis.

- The development of two analytical models: the first uses markov-chain characterization to analyze the additional queuing delay introduced by C-Sec compared to other related protocols. The analysis shows that the additional queuing delay introduced by C-Sec is small and can be tolerated, as most of WSN applications do not require hard real-time constraints [23]. The second analytical model analyzes the effect of packet dependency introduced by C-Sec on the post-decryption error probability compared to other related protocols. Results show that the post-decryption error probability of C-Sec is slightly higher than related protocols. The significance of the energy saving of C-Sec outweighs the small increase of energy consumption due to a slightly higher post-decryption error probability.

- The design of an energy efficient elliptic-curve processor with the least number of mathematical operations, as well as clock cycles per encryption among elliptic curve processors proposed for WSNs in the literature. The proposed processor exploits the parallelism in the XZ coordinate system to reduce number of cycles.

- The design of SN-Sec [24]: a secure sensor node platform that provides energy efficient hardware encryption primitives, based on a modified version of the Zylin Processor Unit (ZPU) [25] by adding new instructions to operate the encryption primitives. The SN-Sec platform design eliminates board level attacks and resists non-invasive and semi-invasive attacks. It also provides secure key storage that is independent for the operating system. SN-Sec components are implemented using VHDL, tested for functional correctness testing using ModelSim, and synthesized for Spartan 6 low power FPGA using Xilinx design suite. Results including hardware area, frequency, time delay, and energy consumption are provided for each component separately and for the integrated system.

1.4 THESIS OUTLINE

The remainder of this thesis is organized as follows:

Chapter 2 provides an overview of the WSN's security. It starts by explaining the security challenges faced by WSN security. Then, it provides a review of different types of security attacks against WSNs and explains their classification based on various approaches. Lastly, it investigates the security services that should be provided to guarantee a decent security level for WSNs.

Chapter 3 starts by providing a brief description of TinyOS, an operating system that is used as a basis for most of the related link-layer encryption protocols. Then, it provides a literature review of the related link-layer encryption protocols proposed for WSNs. A comparison between these protocols is presented, based on the security services provided, modes of operation, encryption primitives used, and their packet formats. In addition, a literature review of encryption primitives implemented in hardware for WSNs is provided. These primitives include symmetric, asymmetric and hashing primitives. Based on this review, recommendations for the selection of encryption primitives for the SN-Sec platform are provided.

Chapter 4 introduces C-Sec the proposed link-layer encryption protocol that reduces energy consumption by eliminating the security related communication overhead. The security services provided by C-Sec and the underlying cryptographic algorithms are also analyzed. To prove the efficiency of the C-Sec, performance evaluations for energy, queuing delay, and error probability based on mathematical models and simulations are presented.

Chapter 5 explains the disadvantages, security vulnerabilities, and the approaches to implementing security platforms for WSNs. It also presents SN-Sec, a secure hardware platform design for WSNs. A detailed description of the designs of the security primitives and other components used for SN-Sec platform is presented. Energy, time and hardware area results using synthesis for Spartan-6 low-power FPGA are presented for each component. Finally, synthesis results for the final design of the SN-Sec platform are presented after integrating all components.

Chapter 6 summarizes the main topics of the thesis and provides conclusions and insights to future research directions.

CHAPTER 2 OVERVIEW OF WSN SECURITY

This chapter overviews' the security of wireless sensor networks, starting by studying the security challenges that face WSN by reviewing the security attacks against WSNs and their classification based on various approached from security perspective. The last section of this chapter investigates the requirements that a security system should provide to guarantee decent security level for WSNs.

2.1 CHALLENGES TO WIRELESS SENSOR NETWORK SECURITY

WSNs have many challenges compared to conventional computer networks. These challenges arise from two main factors: their limited resources and the hostile environments where they are being deployed. Because of these challenges, WSNs are more vulnerable to security threats than traditional networks; as a result, it is not practical to directly use existing security methodologies for WSNs. Hence, to develop adequate security mechanisms for WSNs, it is necessary to recognize and understand the challenges and constraints of WSNs [17]. This is the main objective of this section.

2.1.1 Energy constraints

WSNs are widely accepted because they do not require wired infrastructure. However, this specific feature makes them energy constrained. Once a WSN is deployed, sensor nodes cannot be easily recharged or replaced due to the high operating cost. As a result, the energy stored in the battery must be preserved to lengthen the life of sensor nodes and hence the entire sensor network.

The consequences of this hard energy constraint are the overall topic of this thesis. Of particular concern are principles for energy efficient wireless communication for security related data, and the energy-wise trade-off between computation and radio communication. In addition, the energy consumed for encryption primitives needed to run security protocols is addressed. For example, when a cryptographic function is implemented within a sensor node, the energy cost of the additional security components has to be considered.

Adding security to a wireless sensor network will affect the lifespan of the sensor nodes. The extra energy spent by sensor nodes for security is consumed in a few operations: computations related to essential cryptographic functions like encryption, decryption, signing data, verifying signatures; and the energy required for transmitting the security related data, especially the security data fields added to the headers of each communicated packet; moreover, the energy required to store encryption parameters in a secure way (e.g., secure key storage).

2.1.2 Exposedness of wireless communication

Due to the openness and the exposedness of the wireless medium, an adversary can easily spy on the wireless communication and intercept data messages exchanged between WSN nodes, and then launch attacks based on address spoofing to gain full access to the WSN. In this regard, the proposed C-Sec protocol helps by hiding packet headers, including addresses, for most of the traffic. This helps to defend against attacks that depend on traffic analysis like homing attacks [18].

2.1.3 Unattended sensor nodes

Sensor nodes are usually left unattended for long periods of time; and in some applications for the whole sensor network lifetime. This makes them more exposed to physical attacks than typical computer networks, because they are deployed in an environment that is open to adversaries. This makes it easier to launch attacks, capture nodes and tamper with them to resolve data and security keys, and gives unlimited and unrestricted access to the wireless medium. In addition, being unattended and remotely managed makes it more difficult for sensor nodes to detect and resist physical tampering attacks.

2.1.4 Cost and platform constraints

The cost of sensor nodes has a significant impact on their usage in different types of applications. Lower cost sensor nodes will result in the ability to deploy a network with higher node density, cover more area, and collect more data. However, decreasing the cost will result in limitations in the type, size, and number of components a sensor node can have. It forces the use of a cheap processor, limited memory, and smaller battery. This makes it more difficult to provide processing power or add more components to provide security, because it directly affects the cost of the sensor node.

2.1.5 Intensive mathematical computations for cryptographic primitives

Security depends on encryption primitives. These primitives require intensive mathematical computations. Because of energy, cost, and platform constraints, it became a challenge to implement those primitives efficiently and cost effectively. Researchers follow two approaches to implement them. The first approach is software implementations as in [8-16]. Software implemented encryption primitives overwhelm the weak sensor node processers and engage them in computations that could take several seconds of processing and consume a lot of energy. The other approach is hardware implementations of those primitives such as in [11] [19] [20], and the proposed SN-Sec platform [24]. This approach is more time and energy efficient but it adds more components to the sensor node, which increases the cost.

2.2 Security Threats for WSNs

In this section, security threats against WSNs are identified and a reviewed based on which layer they are launched. In addition, classifications of these attacks are provided based on various approaches in the literature. Finally, security services need to be guaranteed to provide immunity against these threats are identified.

2.2.1 Security Attacks

Many security threats and attacks exist for WSNs. To make it easy to study these attacks and provide adequate defense techniques against them, researches classified these attacks based on different criteria, like the goal of the attack and its results, the methodology of launching it, and its sophistication level. The main categories of WSNs attacks in the literature are presented in this section. Some of these attacks cannot be prevented by the work proposed in this thesis, like destroying nodes, jamming and collision attacks. However, the majority of the presented attacks are based on node compromise and identity theft, as well as the ability to forge or replay messages. These attacks can be defended as the proposed work in this thesis provides immunity against node compromise in software and hardware levels. In addition to the efficient security services provided by C-Sec protocol.

2.2.1.1 Physical Attacks

Destroying nodes: Nodes are usually deployed in remote areas where they are unreachable by the person who deploys them. However, it is assumed that the attacker has unsupervised and unlimited time access to the nodes, and could simply find these nodes and destroy them. These types of attacks are easy to launch, as they do not require any special or expensive equipment. The defense strategy against them is to hide the node by packaging it in a form similar to the surrounding environment, and minimizing or shutting down the wireless communication if they are detected.

Tampering attacks: tampering attacks are a class of attacks that are launched on the physical structure of the node, which requires direct physical access to the node. They can be classified into three main categories: invasive, semi-invasive, and non-invasive attacks [22]. Invasive attacks are sophisticated attacks that take relatively long time and need expensive tools used in semiconductor manufacturing and testing. This requires moving the sensor node from the deployment area to the laboratory where they get access to the internals of chip, examples of such attacks include micro probing and reverse engineering.

Semi-invasive attacks require less time, simpler tools and are usually less expensive than invasive attacks. The access to the chip die is still required but with a non-penetrative attack. An example of these attacks is fault generation with intensive light pulse, radiation, and local heating. Figure 2.1 shows an example of this attack on the PIC16F84 controller. At first, the controller is decapsulated, then the security fuse is located and deactivated using the fault injection attack [22].

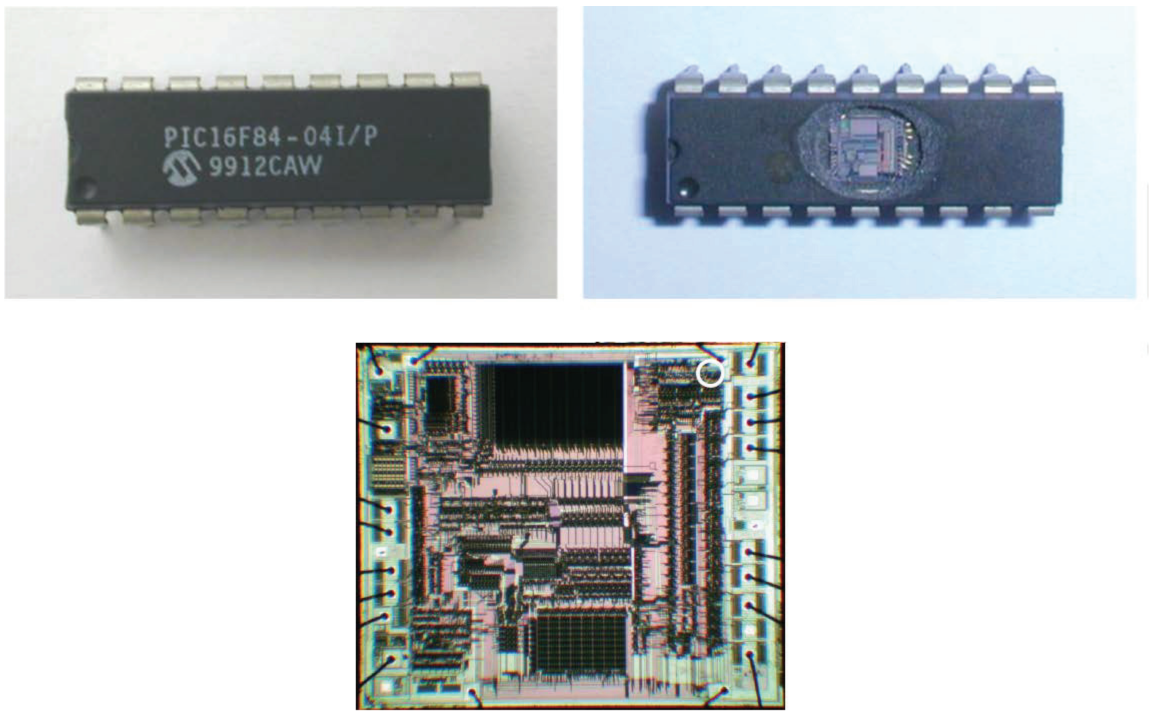


Figure 2.1 A) The PIC16F84 controller B) Decapsulated PIC16F84 controller C) Location of the security fuse in the PIC16F84 controller [22].

In non-invasive attacks, the data is stolen from the sensor node without causing any harm to the attacked device, such as taking advantage of the hardware interfaces of sensor nodes to obtain sensitive data. Non-invasive attacks require previous knowledge of the structure and operation of the chip.

To defend against tampering attacks many precautions can be made. Implementing the processing unit and other necessary hardware designs on Complex Programmable Logic Devices (CPLDs) or FPGAs instead of microcontroller-based devices is recommended, because it is much more expensive to attack and reverse engineer CPLDs and FPGAs. “Even placing important algorithms inside a ‘non-secure’ SRAM-based FPGA makes its reverse engineering very difficult“ [22]. Most of current processing units for WSNs are well known micro-controllers that are not securely designed. They have well-known designs, debugging procedures and hardware interfaces. The information available from their venders facilitates launching non-invasive attacks in a short time and at a low cost. The proposed secure sensor node platform SN-Sec is synthesized for a low power FPGA chip.

Another precaution that increases the cost of reverse engineering is using multilayer Printed circuit Boards (PCBs) and chips in Ball Grid Array (BGA) packages. Such designs make the interconnections of the chips inaccessible and requires special tools and skills to decapsulate the chips. Unmarking the common names and codes of the chips or remarking them with misleading information, repackaging chips, destroying test interface, burning access circuits are also recommended practices that make tampering attacks more difficult [22].

2.2.1.2 Physical layer attacks

Jamming attacks: Jamming attacks address the radio frequencies of the wireless medium used by sensor nodes in order to prevent wireless communication in the network. A jamming attack could be powerful enough to disturb the scope of the entire network or less powerful to disrupt part of it. Jamming can be detected if there is unusual noise during normal operation that leads to degradation in the Quality of Service (QoS), along with low packet delivery ratio and high signal strength levels.

Channel surfing and switching for a good frequency is one solution, but this requires the node transceivers to work in a wide range of communication frequencies, which will increase the node cost. It also has a lot of overhead, especially for flooding the network with communications to follow the frequency band. Generally, sensor nodes are equipped with single-frequency transceivers to maintain low cost and power requirements. This makes them highly vulnerable to jamming attacks.

2.2.1.3 Data link layer attacks

Exhaustion: exhaustion attacks aim to drain the energy of the sensor node by manipulating MAC protocol efficiency. An example of that is continuously sending Request-To-Send (RTS) packets and forcing another node to reply with a Clear-To-Send (CTS) packet and to stay awake expecting a packet. One solution is to encrypt control messages [23], which makes it more difficult to clone them. Putting a limit on the network requests could also help to reduce the effect of exhaustion attacks.

The C-Sec protocol uses the CFB encryption mode to provide confidentiality and semantic security. This mode of operation supports variable block length messages. Hence, C-Sec can encrypt the small-size RTS/CTS packets without increasing their size to the size of the block cipher, which makes it more efficient to defend against exhaustion attacks.

Collision attacks: Collision attacks aim to corrupt packets and cause retransmissions. This is done by sending small packets in a carefully selected timing to create errors in the payloads of legitimate packets, which causes checksum errors and hence retransmissions.

Collision attacks do not consume a lot of energy from the attacking device. However, it causes a lot of interruptions and energy loss in network operation.

Defense mechanisms against jamming attack can also be used to defend against collision attacks. The use of error correcting codes is can also be used, but it can correct limited number of errors and it increases the energy consumption.

2.2.1.4 Network layer attacks

Attacks on routing information: These attacks aim to disrupt traffic in the network by altering, replaying or spoofing routing packets. This could create routing loops, shorten or extends routes, and increase end to end latency. The use of integrity, authentication, and replay protection techniques is capable of defending against these attacks. The proposed encryption protocol C-Sec provides these security services, in addition it hides the packet header fields so that packets carrying routing information cannot be differentiated from other traffic in the wireless medium.

Homing Attacks: The goal of homing attacks [18] is to shut down the entire network by disabling special nodes like cluster heads or nodes responsible for key management. Special nodes can be discovered by analyzing the communication in the wireless medium. Disabling the key nodes can be done by physically destroying them, or by launching any other denial of service attack on them. Homing attacks can be prevented by encrypting packet headers and sending dummy packets to obscure the communication in the network. Hiding the packet headers, including the addresses, the compact mode of the proposed protocol C-Sec provide immunity against homing attacks.

Hello flood Attacks: In these attacks hello packets are transmitted to the whole network by an attacker with high transmission capability. Nodes who receive the Hello packets will consider the attacker as a neighbor and will attempt to transmit packets through it although it could be out of their radio range. Node authentication requires nodes to verify the identity of their neighbors, which helps to defend against hello flood attacks. The SN-

Sec provides energy efficient implementations of encryption primitives that can be used to implement node authentication schemes. Especially schemes based on asymmetric encryption primitives [30-33].

Selective Forwarding: In selective forwarding attacks, the adversary manages to insert a malicious node to the network. This node can selectively forward certain messages and drop others, or drop all traffic and create a black hole attack. This attack becomes more serious if the malicious node is close to the sink i.e. handles a large amount of traffic. To prevent this attack it is important to cover security vulnerabilities in the hardware platform and make it immune to tampering attacks and node compromise, which is done by the proposed SN-Sec platform. In addition, providing adequate authentication, confidentiality, integrity, and replay protection in the security protocol helps to make the node immune against compromise in the software level. These security services are provided by the proposed C-Sec protocol.

If an attack has already happened, the use of malicious node detection techniques to detect then exclude the malicious node from the network, and the use of routing data from multiple paths are good practices to reducing the effect of selective forwarding attacks [34].

Wormhole attacks: a wormhole refers to a low-latency link between two distant parts of the network used to replay messages among these parts of the network convincing nodes from each side that they are neighbors. Figure 2.2 shows an example of wormhole attack, nodes on the two sides of the network communicate through a wormhole extending between node A and node B, instead of communicating through the nodes in the middle of the network. Wormholes change the logical topology of the networks, which affect routing and help to apply other types of attacks on the traffic going through the wormhole. A low-latency link could be established using one or more compromised nodes, and it could be a wired or wireless connection [34].

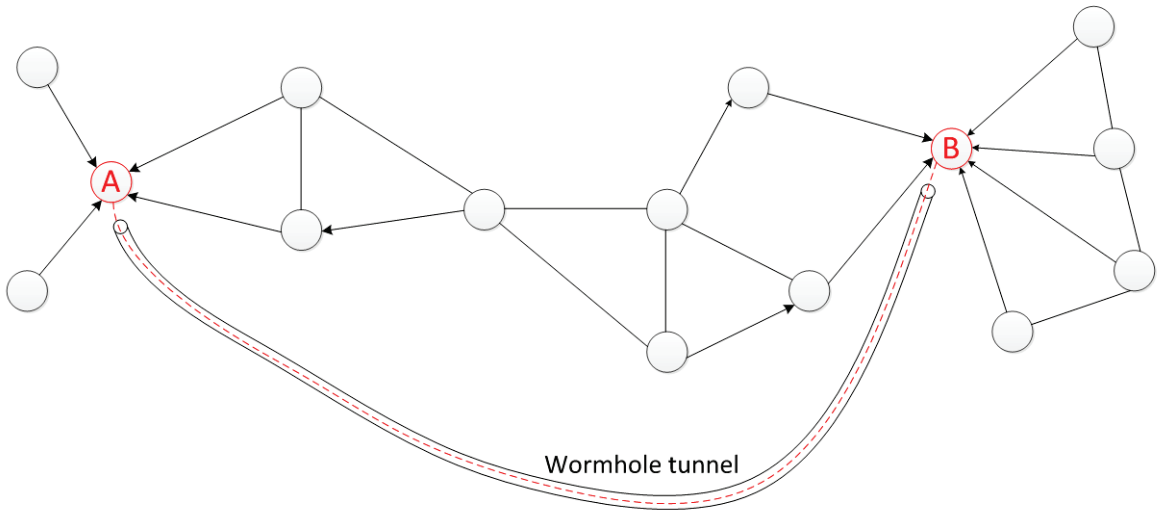


Figure 2.2 Wormhole Attack

Protection against node compromise is the first defense line against wormhole attacks. SN-Sec and C-Sec provides protection against node compromise in the software and hardware levels. Many other mechanisms that depend on the knowledge of the physical location of the nodes were presented in the literature to detect and then defend against wormhole attacks, such as packet leaches [35], which use geographic leaches and temporal leaches to detect wormholes, the use of directional antennas [36], multi-dimensional scaling algorithm [37] and the MAD protocol [38].

Sinkhole attacks: In sinkhole attacks, an adversary tries to change routing information to make a compromised node look more attractive to its neighbor nodes and redirect all possible traffic through it. This will change the logical topology of the network and create a false sink that could be used to launch other types of attacks, like blackholes or selective forwarding attacks. The ability to change routing information and node compromise are the bases for this attack.

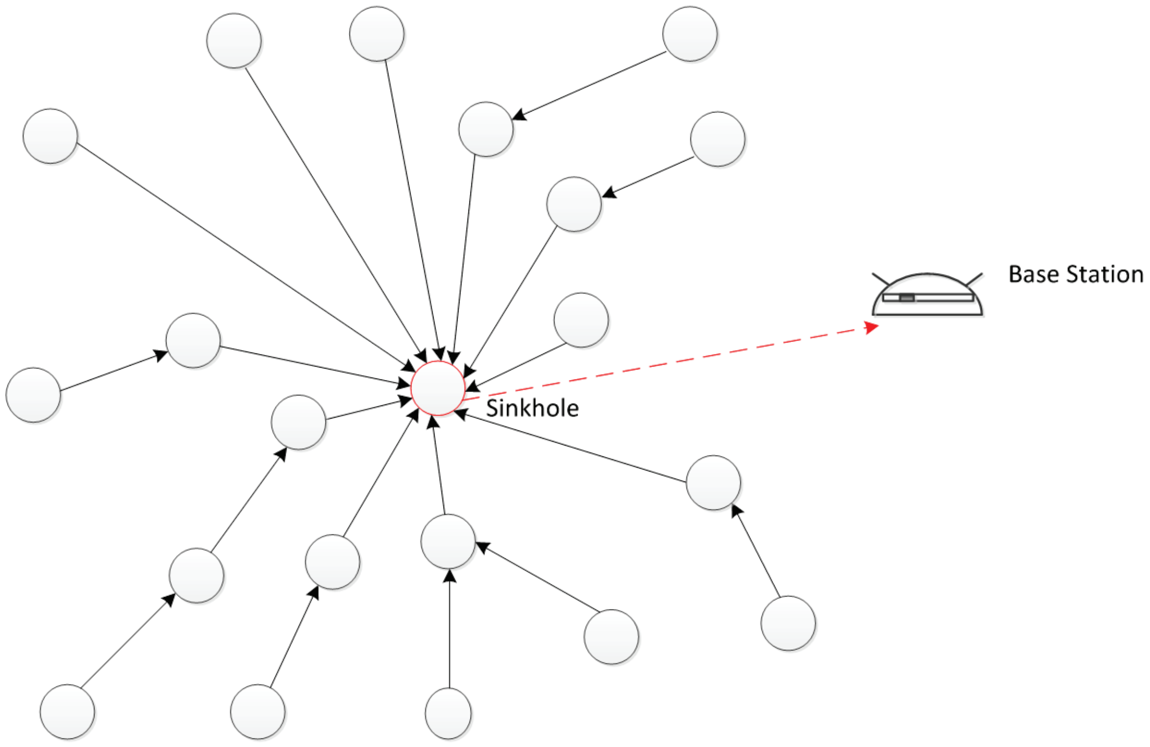


Figure 2.3 Sinkhole Attack

Providing robust security protocols on the software level and robust sensor node platforms against node compromise on the hardware level, which are provided by C-Sec and SN-Sec, can prevent this attack from happening. If the attack has already happened, the same defense mechanisms used to detect and defend against wormhole attacks could be used to reduce its effect [34].

Sybil attacks: In Sybil attacks, the adversary inserts one or more malicious nodes to the network. As Figure 2.4 shows, these nodes use identity fraud to present multiple identities to their neighbors. This disturbs the network topology, and results in consuming node energy and memory in storing information about non-existing nodes. It also affects data aggregation, voting algorithms and many applications that are based on redundancy of data, like fault tolerant algorithms and reliable distributed storage; or applications that aim to eliminate redundancy of data to save power. In both cases, the only copy of the data could be stored with a malicious node under single or multiple identities.

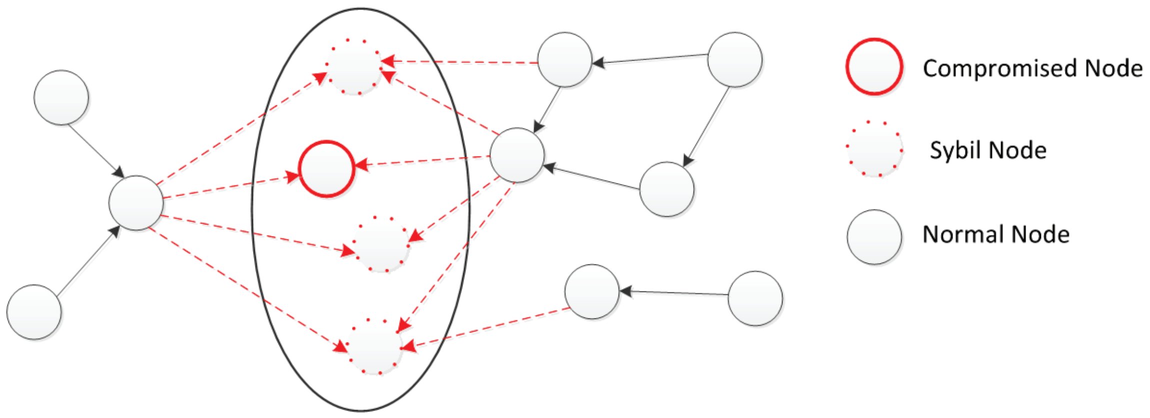


Figure 2.4 Sybil Attack

Node compromise and identity theft are the bases for this attack. Robust security protocols with strong authentication algorithms and immunity against node compromise in the software and hardware levels are the best defenses against Sybil attacks [39]. These services are provided by the proposed C-Sec protocol and the SN-Sec platform.

2.2.1.5 Transport layer attacks

Flooding: flooding attacks target protocols that require maintaining the state of end-to-end connections between nodes. An adversary uses compromised nodes to flood the network with requests to create new connections with the aim of exhausting the storage, processing, and energy resources of sensor nodes, and causing legitimate connection requests to be ignored [30].

Immunity against node compromise in the software and hardware levels, which are provided by C-Sec and SN-Sec, can prevent this attack. If the attack has already detected, the use of client puzzle [41] is a defense mechanism against flooding attacks. In this mechanism, the server node proposes solving a computationally expensive puzzle to the client nodes in order to accept the connection. This keeps the attacking node busy, and hence, reduces the connection request rate and protects legitimate node resources from vanishing. Although puzzle solving has its own processing power overhead on legitimate nodes as well, it is considered much better than wasting communication energy on fake connection requests.

Desynchronization: In this attack, the adversary targets synchronization procedures in existing connections between legitimate nodes. This can be done by sending fake retransmission requests or spoofing messages with faulty sequence numbers. This wastes node energy in retransmission and costly synchronization procedures and reduces communication ability between nodes. The best solution to prevent desynchronization attacks is to use strong authentication, confidentiality, integrity, and replay protection techniques in the security protocol between nodes [30]. These services are provided by the proposed C-Sec protocol.

2.2.1.6 Application layer attacks

Overwhelm Attack: In the overwhelm attack [32], an attacker creates a large amount of stimuli by injecting forged or replayed query messages. This attack is so powerful because a single stimulus message broadcast over the network will force all sensor nodes who receive this message to report sensor measurements to the base station. This results in forwarding a large volume of traffic to the base station, which consumes network bandwidth and energy.

This attack depends on the ability to forge or replay query messages. Efficient replay protection, confidentiality, integrity, and authentication algorithms and high resilience to node compromise can prevent it. These services are provided by the proposed C-Sec protocol and the SN-Sec platform. In addition, this attack can be prevented if sensor readings are reported at fixed time intervals rather than data query requests. Reducing the volume of network traffic can help in reducing the effect of this attack once it happens. This can be done by rate limiting and data-aggregation algorithms [32].

Path-Based DoS (PDoS): In PDoS attacks [43], the adversary overwhelms nodes on the path to the base station by flooding their multi hop end-to-end communication channel with either replayed or forged packets. As Figure 2.3 shows, this will exhaust the nodes along the targeted path; nodes downstream will not be able to communicate with the base station. This attack depends on the ability to forge or replay query messages. Efficient replay protection, integrity, and authentication algorithms and high resilience to node compromise can prevent PDoS attacks. These services are provided by the proposed C-Sec protocol.

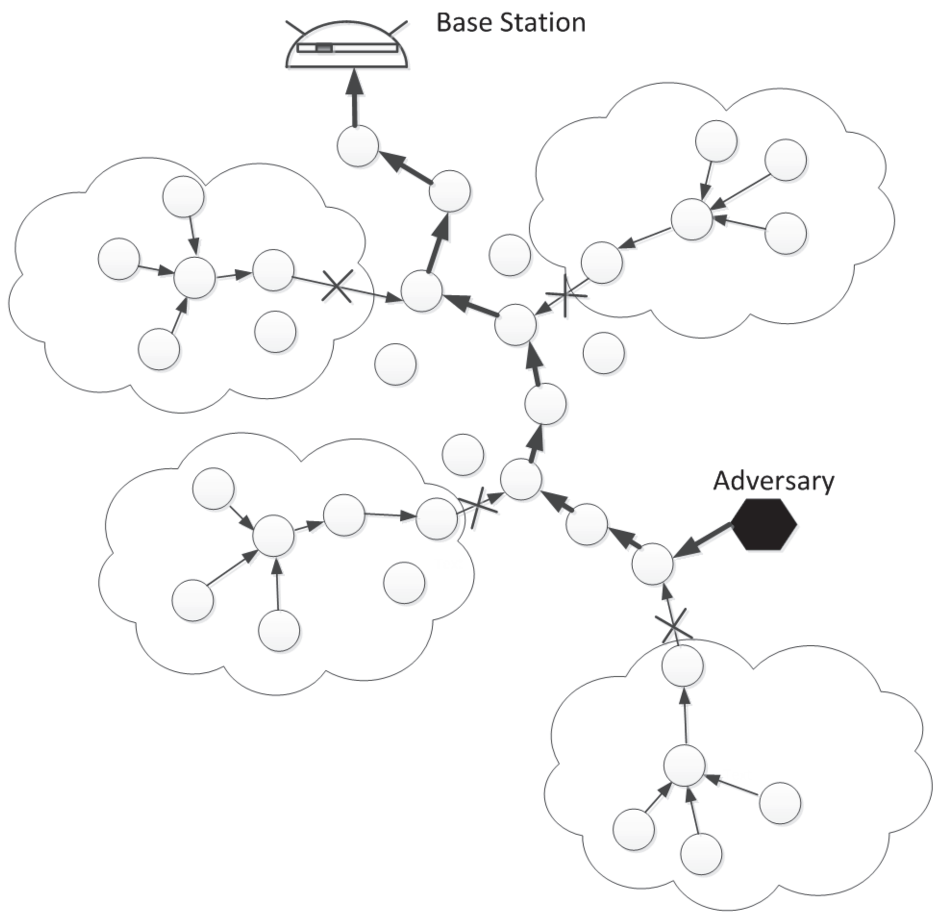


Figure 2.5 PDoS Attack

Network programming attack: In-network programming is a useful feature that makes it easy to remotely update the code running on sensor nodes. Network programming attack addresses this feature. The adversary launches the attack by injecting a false program to the node through forged packets or compromised nodes. The injected code could lead to losing control of the network or returning altered sensor readings or sending data reports to the adversary. Strong authentication schemes as in the hash-chaining technique [44] where the program is divided into parts, and each part has hash of next part. The proposed C-Sec protocol provides a hash-chaining scheme; each transmitted packet has the hash code of the previous packet. This makes it more robust against network programing attacks.

2.2.2 Classification of attacks

The classification of the attacks based on various criterion helps to understand them and develop defense mechanisms against them. This section provides a description of the various classifications of attacks on WSNs in the literature.

2.2.2.1 Attacks on availability

Also known as ***Interruption*** or ***Denial of Service (DoS)*** attacks. The main goal is to stop or disturb the normal operation of the sensor network or exhaust any of its resources like its energy, wireless bandwidth, processing power, and memory. The result of this could make the sensor network or part of it unavailable permanently of for a period of time, disturbing its functionality and operations and shortening its life time. The unattended operation, limitations and constraints of WSNs make them vulnerable to a wide range of DoS attacks that can be launched on different layers.

Destroying nodes, jamming attacks, exhaustion, collision attacks, wormhole, sinkhole, Sybil, flooding, and desynchronization can be classified as DoS attacks because they disrupt the operation of the WSN and reduce its lifetime.

2.2.2.2 Attacks on Secrecy

Also known as ***Interception*** attacks**.** The goal of these attacks is to get secret information from the WSN. This information could be related to the data collected and computed by sensor nodes or data related to the operation of the network, like the routing information and encryption keys. If interception does not result in any changes in the data or operation of the network, it is called ***Passive attacks***, such as eavesdropping on communication.

2.2.2.3 Attacks on Integrity

Also known as ***Modification*** attacks. These attacks result in changes in the data communicated within the network. These changes could be in the measured date reported by sensor nodes or in control data. That could result in changes in the topology, operation, and functionality of the sensor network.

2.2.2.4 Attacks on Identity

Also known as ***Fabrication*** attacks. These attacks include violation of the authenticity of communicated data. This is usually done by injecting new packets to the network. These packets could carry wrong information about the phenomena being watched or control packets such as exhaustion, hello flood, and desynchronization attacks.

2.2.2.5 Active attacks

Active attacks results in changes in the data or behavior of the WSNs. This includes Interception like dropping packets in selective forwarding attack, as well as Modification and Fabrication as in exhaustion, hello flood, and desynchronization attacks.

2.2.2.6 Internal attacks.

Internal attacks are launched by a compromised node that belongs to the WSN. Internal attacks are more dangerous because it is hard to detect them. In addition, the attacking node may have secret information about the network, like the encryption keys. Also it could have the trust of other sensor nodes because it was originally a legitimate node.

2.2.2.7 External attacks

External attacks are launched by devices that do not belong to the WSN. These attacks can be classified into two classes based on the strength of the attacking device. ***Laptop class*** attacks where the attacking devise is powerful, and ***Mote class*** attacks where the attacking device is a usual sensor node.

Table 2.1 provides a summary of the classification of WSNs security attacks discussed in this section.

Table 2.1 Classification of attacks on WSNs

	Layer of operation	**Attack on Availability (DoS)**	**Attack on secrecy**	**Attacks on Identity**	**Passive /Active**	**Internal /External**	**Node class /Laptop class**
Destroying nodes	Physical	Yes	No	No	Active	N/A	N/A
Jamming attacks	Physical	Yes	No	No	Active	external	Laptop class
Exhaustion	Data link	Yes	No	No	Active	Both	Both
Collision attacks	Data link	Yes	No	No	Active	Both	Both
Eavesdropping	All	No	Yes	No	Passive	Both	Both
Attacks on routing information	Network	Yes	Yes	Yes	Active	Both	Both
Traffic Analysis	Network	No	Yes	No	Passive	Both	Both
Homing attacks	Network	Yes	Yes	Yes	Active	Both	Both
Hello Flood	Network	Yes	No	Yes	Active	External	Laptop class
Selective Forwarding	Network	Yes	No	No	Active	Both	Both
Wormhole attacks	Network	Yes	Yes	Yes	Active	Both	Both
Sinkhole attacks	Network	Yes	Yes	Yes	Active	Both	Both
Sybil attacks	Network	Yes	Yes	Yes	Active	Both	Both
Flooding	Transport	Yes	Yes	Yes	Active	Both	Both
Desynchronization	Transport	Yes	Yes	Yes	Active	Both	Both
Overwhelm	Application	Yes	Yes	Yes	Active	Both	Both
Network programming attack	Application	Yes	Yes	Yes	Active	Both	Both
PDoS attack	Application	Yes	Yes	Yes	Active	External	Laptop class

2.3 Security Requirements for WSNs

As mentioned in the previous sections, WSNs have many limitations and constraints that make them vulnerable to various types of attacks. To prevent these attacks or minimize their effect on WSNs, security mechanisms are required to provide security services. These services are:

Confidentiality: Confidentiality ensures that the data transmitted between communication parties is not disclosed to unauthorized users. This can be achieved by the use of encryption, with the encryption keys shared only between authorized users.

Semantic Security: Semantic security is a higher level of confidentiality which ensures that the adversary cannot learn anything about a plaintext from its ciphertext. This is usually achieved by using block cipher modes of operation that support semantic security.

Integrity: Data integrity helps the receiver to ensure that the received data is not altered by an adversary during transmission. Integrity can be accomplished by sending a message digest that is impossible to reverse along with the message. This digest is produced using a one-way hash function that combines all the bytes of the message with a secret key only known to communication parties.

Authentication: Data authentication allows verifying that the data is sent by the claimed sender. This is important to create a barrier between external and internal members of the network. Without authentication, any attacker could claim to be a member of the network. Authentication is accomplished by sending a message digest that is impossible to reverse along with the message. This digest is produced using a one-way hash function that combines all the bytes of the message with a secret key only known communication parties.

Non-repudiation: aims to ensure that the transmitted message has been sent and received by the communication parties claiming to be the actual sender and receiver of that message. This guarantees that neither of them can later deny sending or receiving the message. Non-repudiation is usually achieved by the use of digital signature.

Data Freshness and replay protection: Data freshness aims to ensure that the transmitted messages are new and to prevent an adversary from replaying old messages. This is usually achieved by explicitly sending a monotonically increasing freshness counter with the message, Or by maintaining this counter on both sides of communication and in the computation of the message digest.

Availability: aims to ensure that the network resources are accessible and functional whenever they are needed. The loss of availability caused by an attack on the network or one of its resources is referred to as Denial of Service attack. Providing availability is difficult for WSNs because of unattended operation and its limited resources.

2.4 CONCLUDING REMARKS

This chapter provided an overview for the security of WSNs, starting by studying the challenges that face WSN security, then reviewing the security attacks against WSNs and their classification based on various approached from a security perspective. Some of these attacks cannot be prevented by the work proposed in this thesis, like destroying nodes, collision, and jamming attacks. However, the majority of the presented attacks are based on node compromise and identity theft, as well as the ability to forge or replay messages. These attacks can be defended as the proposed work in this thesis provides immunity against node compromise in software and hardware levels, in addition to the efficient security services provided by C-Sec. This chapter also investigated the essential requirements that a security system should provide to guarantee adequate security level for WSNs. As will be discussed in detail in Chapter 4, these requirements are provided by C-Sec protocol.

CHAPTER 3 RELATED WORK

In this chapter, link layer encryption protocols proposed for WSNs are examined and compared. This comparison includes the security services provided, modes of operation, encryption primitives used and their packet formats. Most of the related protocols are based on the TinyOS operating system. A brief description of TinyOS and its packet format is presented in section 3.1. The comparison between WSN link-layer encryption protocols is presented in section 3.2. In section 3.3, a review of encryption primitives implemented in hardware for use of WSNs is presented. These primitives include symmetric, asymmetric and hashing primitives. Based on this review, recommendations for the selection of encryption primitives for the SN-Sec platform are provided.

3.1 THE TINYOS

The design of operating systems for WSNs is different from traditional operating systems, due to the significant challenges of WSNs including sparse energy, cost limitations, constrained processing and memory resources, and unattended operation. These challenges impose special requirements for WSN operating systems, such as flexibility, small kernel code footprint, runtime configurability of services, and reprogramability. It also requires efficient execution model services like synchronization, scheduling, and communication between system components. In addition, the operating system should have clear Application Programming Interfaces (APIs) including networking, sensing, memory management, power management, and process management APIs [45].

TinyOS [36] is an open source WSN operating system widely used with different types of commercially available sensor nodes known as “motes”. TinyOS can be easily adapted with different types of hardware; this made it the best choice for various industrial applications and academic research. It uses a component-based programming model coded in nesC language [37] . This language supports complicated concurrent programming with extremely limited resources and low memory requirements. The code size of the basic core of the TinyOS is only 400 bytes. In addition, most of the applications can fit with less than 16KB of memory, which is very efficient and facilitates low-power operation.

The TinyOS scheduler follows a non-preemptive First In First Out (FIFO) policy. Although this policy simplifies the handling of deadlocks, race conditions, and other concurrency-related, it could result in serious problems in many cases, especially with applications with long running tasks. These issues could significantly reduce the responsiveness of the system and result in delaying priority tasks [38].

TinyOS uses synchronous acknowledgments protocol on the link level; a synchronous communication protocol requires a response for each transmission before it starts the next transmission. Only one transmission can be "on the flight" at a time. This is also known as stop-and-wait or alternate-bit Automatic Repeat Request (ARQ). This mechanism is used with most of the WSN operating systems [49] [40]. Because WSN applications require low power as well as low bandwidth communication, and can resist moderate packet latency.

The main source of power consumption is communication compared to computation and sensing [41]. The high energy-cost of communication in WSNs imposes smaller packet sizes, since that is optimal for communication-energy efficiency [42]. The TinyOS frame format is shown in Figure 3.1. This format will be used as a baseline reference for the comparison between the related encryption protocols. The TinyOS was used as a base for most of these protocols. In addition, it is a widely used for WSN applications in general. This frame has a maximum size of 43 bytes: 13 bytes of header and trailer fields, and a maximum of 30 bytes of payload. The basic header fields are: a 6-byte preamble, 2-byte destination address (*Dest)*, a one byte Active Message *(AM)* field which plays a role like the port number in TCP/IP networks, a one byte length field *(Len)*, a one byte group field *(Grp)* used as a unique network identifier to distinguish from other networks, and finally a Cyclic Redundancy Check *(CRC)* trailer field.

Preamble	Dest	AM	Len	Grp	Payload	CRC
6 bytes	2 bytes	1 byte	1 byte	1 byte	0 - 29 bytes	2 bytes

Figure 3.1 TinyOS frame

Considering the small size of the TinyOS frame, the size of the header is relatively large. Increasing the header size by adding more fields to implement security services will dramatically increase the energy needed to transmit the packet. The proposed C-Sec protocol eliminates the need to transmit all header fields related to security most of the time.

3.2 Overview of WSN encryption Protocols

Link layer security is of a unique importance in Wireless Sensor Networks (WSNs). Unlike traditional networks, similar data traffic is generated in nodes throughout the network and addressed to one sink. Because this traffic has a high level of similarity, sensor nodes aggregate it to minimize communication volume and save energy. The end-to-end security schemes in traditional networks are not practical for WSNs, because the data has to be inspected and aggregated on the way to the sink. In addition, link layer based security minimizes the effect of security attacks. Security attacks can be discovered on the next hop once they happen, but can only be discovered at the sink if end-to-end security schemes are used.

Many link-layer security protocols were recently proposed for WSNs [8-16]. The design of those protocols is challenging due to the constrained nature of WSNs that restricts reserving additional resources to provide security services. These resources can be in the form of energy consumption, processing overhead, additional hardware components, and communicated data.

This section reviews the most recognized encryption protocols designed to implement the basic security services for WSNs in the literature. From a security prospective, the protocols are compared based on the security services they provide. Failing to provide all security services leaves vulnerabilities from which a WSN can easily be attacked. The security of the encryption and MAC algorithms used has large impact on the security services provided. For example, the use of 80-bit encryption is vulnerable to brute force attacks and CBC-MAC has security weaknesses for variable length messages.

From the energy efficiency prospective, the amount of header size overhead they add to each packet is the main factor. The designers of related protocols tried to minimize the amount of packet header overhead related to security in many ways; nevertheless, all of them added fields to the headers of each packet to implement security services. In addition, the computational complexity of the encryption and MAC algorithms used, its implementation methodology and mode of operation plays important role in energy efficiency.

3.2.1 Zigbee security

Although Zigbee protocol suite [11] is designed for wireless Personal Area Networks (PANs), it is being used for wide range of applications that require low data rates (i.e. 20 to 250 kbps), long battery life, and secure networking. Zigbee is widely used for WSNs because of its cheap price and efficient implementations. Most of its implementations include hardware modules for encryption, which is computationally efficient.

Zigbee provides many security services. It provides authentication and confidentiality by encrypting the packet payload with the AES-CCM (Counter with cipher block chaining message authentication code) mode, which is a special NIST standard [43]. As Figure 3.2 shows, new fields were added to the packet to implement the required security services. These fields include A two-byte source address field *Src* (instead of six-byte IEEE address used with PANs), a four-byte Message Authentication code field *MAC*, which has the minimum size of MAC supported by the AES-CCM standard.

Preamble	Dest	Src	AM	Len	Ctr	Payload	CRC	MAC
6 bytes	2 bytes	2 bytes	1 byte	1 byte	4 bytes	0 - 29 bytes	2 bytes	4 bytes

Figure 3.2 Zigbee frame

A four-byte counter field *Ctr* is also added to guarantee frame freshness and replay protection. Notice that the group field *Grp* of the TinyOS frame was removed to save energy. The functionality provided by the group field is implicitly implemented with the keying mechanism associated with the MACs. This mechanism provides access control and is sufficient to distinguish between networks.

Although Zigbee security design provides all of the basic security services, the additional security related frame size overhead is considered high. The removal of the *Grp* field saves one byte. However, the size of other additional fields related to security is 10 bytes (4 *MAC* + 4 *Ctr* + 2 *Src*). The Zigbee frame size is quite large compared to the plain TinyOS frame, which has a maximum size of 43-bytes.

3.2.2 TinySec

TinySec [10] is a link-layer security architecture that was implemented as part of the TinyOS operating system. It provides tradeoffs between performance and security by having optional two-level security modes. The TinySec-Auth mode only provides message integrity and authenticity. As Figure 2.3-a shows, TinySec-Auth mode adds a four-byte *MAC* field to the frame; this field is computed using the CBC-MAC produced by running the Skipjack [54] algorithm over the packet header and payload.

While the TinySec-AE mode provides message confidentiality and replay protection in addition to the message integrity and authenticity provided by the TinySec-Auth mode. This is done by encrypting the payload, and adding a two-byte freshness counter field *Ctr* to the packet to provide replay protection. The initial vector of the CBC mode of the encryption algorithm consists of a concatenation of the *Dest*, *AM*, *Len*, *Src*, and *Ctr* fields. The Skipjack algorithm was chosen for TinySec because of its smaller software implementation code size compared to other encryption algorithms. It produces a 64-bit block size using an 80-bit encryption key. This key size has been considered unsecure against brute force attack since 2012 [55].

Neither of TinySec modes includes the *Grp* field as in the Zigbee security protocol. In addition, the *CRC* field is removed. Its functionality is implicitly implemented with the *MAC* field. Since MACs are used to detect malicious changes in communicated packets, they can also be used to detect transmission errors as a replacement for the CRC.

Figure 3.3 shows the frame of both of the TinySec modes. The TinySec-Auth mode adds only four bytes for the MAC field; with the removal of the *Grp* and *CRC* fields, these results in a 44-byte frame, which is only one byte larger than the TinyOS plain frame. However, the level of security that TinySec provides is inadequate, because it only provides packet authentication.

The TinySec-AE mode is used for comparison with other protocols because it provides all the basic security services. The TinySec-AE mode adds 8 bytes to the basic TinyOS frame (4 *MAC* + 2 *Ctr* + 2 *Src*). The elimination of the *Grp* and *CRC* fields saves 3 bytes. The final frame is 5 bytes larger than the TinyOS frame.

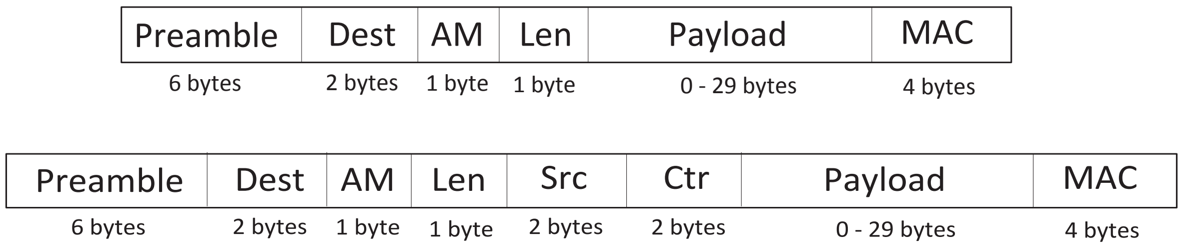


Figure 3.3 a) TinySec-Auth packet. b) TinySec-AE packet

3.2.3 Secure Network Encryption Protocol

The Secure Network Encryption Protocol (SNEP) was proposed as part of SPINS [8]; a security protocol suite for WSNs. It aims to provide data confidentiality, two-party data authentication, and data freshness, with low overhead. SNEP uses RC5 (Rivest Cipher 5) [47] in the counter mode to provide data confidentiality through encryption. RC5 is a 64-bit block, 128-bit key encryption algorithm. It was chosen over the AES because of its small code size for software implementations. It also uses the CBC-MAC mode of the RC5 to produce the MAC and provide authentication and integrity.

The SNEP protocol provides two levels of freshness. Weak freshness is automatically provided by the RC5 encryption in the counter mode. Since the sender increases the counter after sending each message, the receiver confirms weak freshness by verifying that received messages have a monotonically increasing counter. In strong freshness, the sender node creates a 64-bit random number and sends it in a request message to the receiver. The receiver generates a response message and uses this number in the computation of the *MAC* field. If the *MAC* of the response message verifies successfully, the sending node recognizes that the response was generated by the receiver node and hence strong freshness is achieved.

The SNEP protocol does not include the *Grp* and the *CRC* fields. It replaces their functionalities with existing fields as in TinyOS. For further reduction in the header size, the SNEP protocol updates the freshness counters on both sides of communication and does not transmit them, which saves another two bytes. However, it uses an 8-byte MAC field that is twice as large as other protocols.

Although 8-byte MACs are more secure, it is considered expensive for a sensor node to transmit them in the wireless medium. All of the related protocols used 4-byte or less for that purpose. With a 4 byte MAC, the attacker has 2^{32} choices to try to get a valid MAC for a particular frame. The average success rate for that is 2^{31} tries. In other words, an attacker has to send a forged packet to the receiver 2^{31} times before it can succeed to pass one forged packet. For low bit rate WSNs, sending 2^{31} false packets will require occupying the wireless medium for a very long time, which makes it easily discoverable. In addition, the batteries of the sensor nodes will drain before it can receive such a large number of messages.

Figure 3.4 shows the format of the SNEP frame. SNEP saves sending 5 bytes in the header i.e. the *Grp*, *CRC*, and *Ctr* fields, but it requires sending a two-byte source address *Src* and a lengthy 8-byte *MAC* filed. This makes it 7-bytes larger than the TinyOS plain frame.

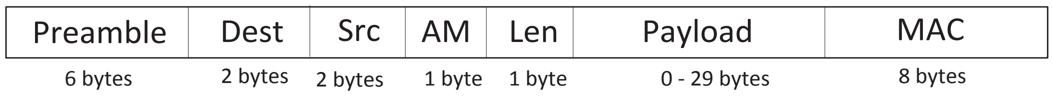


Figure 3.4 SNEP packet

3.2.4 MiniSec

MiniSec [48] uses the Offset Code Book (OCB) [49] encryption mode, which combines authentication and confidentiality in one pass. This saves running the encryption algorithm twice for encryption and then authentication as in other related protocols. MiniSec uses software implementation of Skipjack algorithm with a 64-bit block size and an 80-bit key. The designers of the MiniSec protocol suggested to switch to AES algorithm when Skipjack becomes unsecure in 2012 [55] .

MiniSec has two modes of operation. MiniSec-U mode is used for unicast communication and MiniSec-B mode for broadcast communication. The MiniSec-U tries to strike a balance between two extremes, sending the whole freshness counter as in TinySec and not sending it at all but updating it on both sides of communication as in SNEP. The drawback of the TinySec approach is that it consumes extra communication energy by sending 16 more bits in the wireless medium. The SNEP approach suffers from a costly counter resynchronization problem in case the counter is wrongly updated because of dropped packets. MiniSec-U keeps a synchronized increasing counter between communication parties. However, it only sends the least *x*-bits of the counter with each packet. By keeping *x* low (i.e. 3-bits), the energy consumption for communication is kept almost as low as not sending the counter at all.

MiniSec-B mode addresses broadcast communication. It is expensive to run the counter resynchronization protocol among many receivers. In addition, if a node simultaneously receives packets from a large group of sender nodes it has to maintain a freshness counter for each, resulting in high memory demand. MiniSec-B addresses these issues through two approaches. The first is a sliding window approach where the time is divided in to epochs. The receiver ignores packet from older epochs, which limits the window of the replay attack to one epoch.

The second approach uses Bloom Filters (BFs). A Bloom Filter is “a space-efficient probabilistic data structure that is used to test whether an element is a member of a set” [50]. All nodes maintain two alternating BFs for the two most recent active epochs. All of the received packets are stored in the associated BFs. The receiving node accepts the packet if it does not belong to any of the BFs. If the packet is found in any of BFs, then it is dropped because it is a replayed packet.

The MiniSec protocol uses the Most Significant Bits (MSB) of some header fields to overload the freshness counter and save packet size. As shown in Figure 3.5, the MiniSec-U mode provides weak replay protection through a 3-bit counter stored in the MSBs of the length field. The MiniSec-B mode has a higher level of security; it uses an 8-bit freshness counter by adding the other 5-bits of the counter in the place of the five MSBs of the destination address. However, this will reduce the address space to around 2k addresses instead of 64k addresses. Both MiniSec modes have a maximum frame size of 46 bytes. This is 3-bytes larger than maximum frame size of TinyOS.

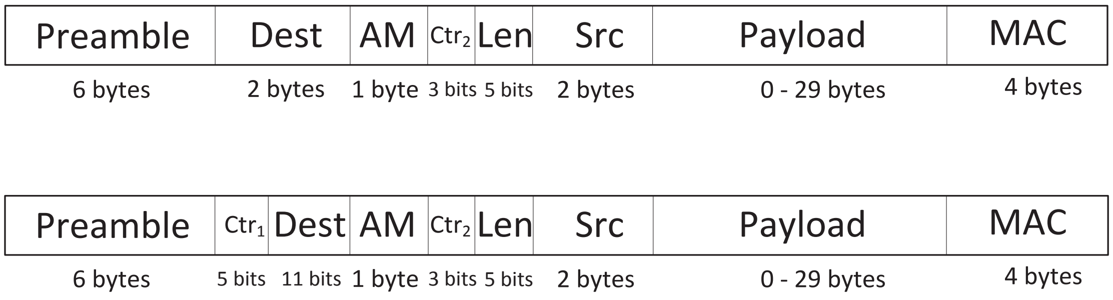


Figure 3.5 A) MiniSec-U packet. B) MiniSec-B packet

3.2.5 Link Layer Security Protocol

The Link Layer Security Protocol (LLSP) [12] uses AES encryption in the CBC mode to provide message confidentiality, and CBC-MAC to provide message authentication. It also maintains a 4-byte counter on both sides of communication to provide replay protection. To save communication energy the counter is not transmitted, and maintain on both sides of communication. The initial vector of the CBC-MAC consists of a combination of the *Dest*, *AM*, *Len*, *Src*, and *Ctr* fields. Figure 3.6 shows the LLSP frame format, it is similar to the SNEP frame except that the MAC field is 4-bytes instead of 8-bytes. The resulting frame size is 46-bytes. This is 3-bytes larger than the maximum size of the TinyOS frame.

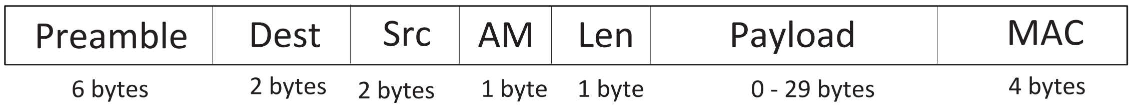


Figure 3.6. LLSP frame format

3.2.6 SenSec

The SenSec [52] defines a variation of the Skipjack algorithm called Skipjack-X. It also defines the CBC-X mode of operation to provide confidentiality and authentication in one pass and provide a padding technique that supports variable size messages. The SenSec keeps the *Grp* field in the original TinyOS format and uses it as a part of the initial vector of the CBC-X mode, which is defined as a concatenation of *Dest*, *AM*, *Len*, *Grp*, and *Ran* fields, where *Ran* is a random number sent with the frame header. The first *Ran* is generated by encrypting the first packet header with the *Ran* field set to zero. After that, the three least significant bytes of the ciphertext are used to produce it. For the next packets, the 3-byte *Ran* field is determined from the three least significant bytes of the MAC of the previous frame. Figure 3.7 shows the frame format of the SenSec protocol. It is 5-bytes larger than the TinyOS frame.

Preamble	Dest	AM	Len	Grp	Ran	Payload	MAC
6 bytes	2 bytes	1 byte	1 byte	1 byte	3 bytes	0 - 29 bytes	4 bytes

Figure 3.7 SenSec frame format

3.2.7 SecureSense

SecureSense [11] is a link-layer security framework that provides different levels of security services, depending on external environments situation, internal constraints and application requirements. It uses RC5 in the CBC encryption mode and CBC-MAC to produce the MAC.

SecureSense uses an eight-bit Security Composition Identifier (SCID) to specify the security level. Four out of the eight bits are used to determine which of the four security levels is used. The security levels are confidentiality, integrity, semantic security with implicit counter and replay protection. Two bits are used to determine the strength of the cipher and the other two bits are reserved for future use.

The SecureSense replaces the *AM* field with *SCID* to represent the type of security composition for each packet; similar to TinySec, the *Grp* field is removed in SecureSense since its functionality can be replaced with the keying mechanism. Either the *MAC* or the *CRC* fields are used in the frame if the integrity and access control services are required.

Figure 3.8 shows the frame formats of the SecureSense protocol. The size of the frame for the mode that provides the maximum-security level including confidentiality, integrity, semantic security, replay protection, is 50 bytes, 7-bytes higher than the maximum size of the TinyOS frame.

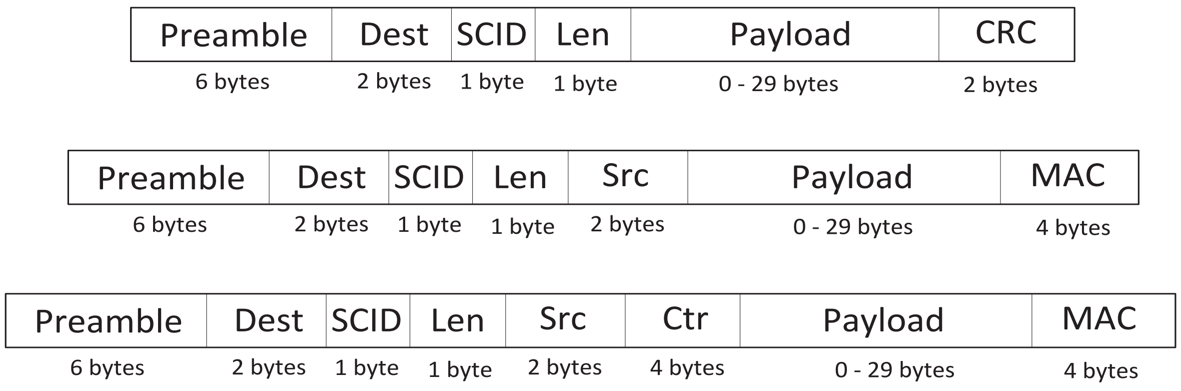


Figure 3.8 SecureSense frame formats A) Only confidentiality. B) Without semantic security. C) Maximum security level

3.2.8 FlexiSec

The FlexiSec [12] protocol aims to provide a secure communication stack that is configurable depending on the data rate, level of security desired and the security attributes desired based on the nature of the application. It uses either XXTEA [53] or AES ciphers – depending on the available resources. It also uses the OCB authenticated encryption block cipher mode of operation instead of CCM or Galois Counter Mode (GCM) [54] because OCB mode is more energy efficient for software implementation. Bloom filters are used for authentication when encryption is not needed.

FlexiSec has nine modes of operation; the first mode is no security or security dependent on hardware. FlexiSecHASH mode provides message integrity with un-keyed authentication without any demands for the entity authentication. FlexiSecAUTH64 provides support for 8-bytes of MAC but without data encryption. FlexiSecAUTH32 offers authentication-only support for low data rate applications – with the provision of only 4 bytes of MAC. FlexiSecAUTH_ENC64 and FlexiSecAUTH_ENC32 are the options to support data confidentiality apart from the message integrity using 8-byte and 4-byte MACs respectively.

The FlexiSecAUTH_REPP64 and FlexiSecAUTH_REPP32 modes extend the security properties in the previous modes with of replay protection. However, these modes could be employed for applications demanding message authentication as well as replay protection alone, without any encryption. Therefore, these modes are intended to employ CBC-MAC as the message authentication scheme. The FlexiSec_AUTH_ENC_REPP64 offers all the security attributes listed above with an 8-byte MAC using the OCB mode.

Figure 3.9 shows the different packet formats of the FlexiSec protocol. The FlexiSec_AUTH_ENC_REPP64 mode is the only mode that provides all the security services. Its header size is 20 bytes; this makes it 7-bytes larger than the TinyOS header.

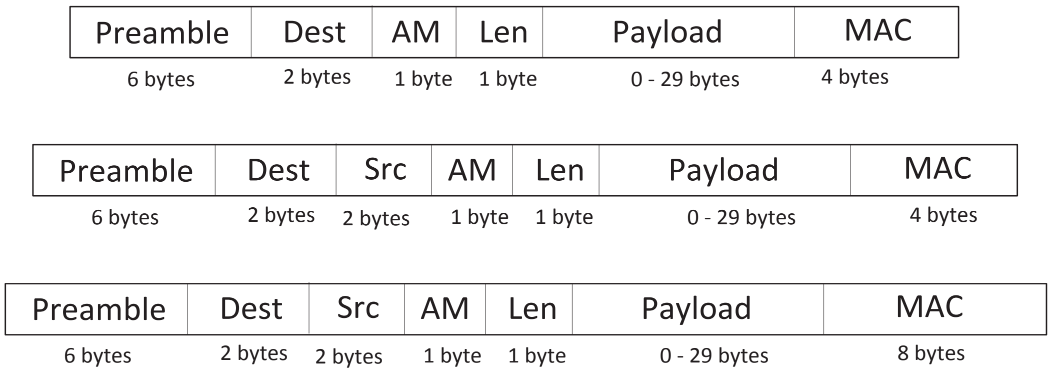


Figure 3.9 FlexiSec frame formats: A) FlexiSecHASH B) FlexiSecAUTH32/ FlexiSecAUTH_REPP32, C) FlexiSecAUTH64, FlexiSecAUTH_REPP64, FlexiSec_AUTH_ENC_REPP64

3.2.9 WSNSec

WSNSec [55] uses Scalable Encryption Algorithm (SEA) in the counter mode, and CBC-MAC to provide semantic security and authentication. The SEA algorithm is used because it has 192-bit data-block and key size, with a small increase on the memory usage and energy consumption compared to its counterparts. Figure 3.10 shows the frame format do the WSNSec protocol. This format is very similar to the TinySec frame format. The only difference is that the 4-byte MAC field is encrypted with the payload field for more security. The WSNSec frame is 5 bytes larger than the TinyOS frame.

Preamble	Dest	AM	Len	Src	Ctr	Payload	MAC
6 bytes	2 bytes	1 byte	1 byte	2 bytes	2 bytes	0 - 29 bytes	4 bytes

Figure 3.10 WSNSec frame format

3.2.10 Comparison of related works

Table 3.1 shows the header fields for the related protocols and the proposed C-Sec protocol. The compact mode of the C-Sec protocol, which is used most often, has the smallest header size of all related protocols. It is even 3-byte smaller than the TinyOS plain header that does not include any fields to implement security services. Meanwhile, it provides all the required security services at the MAC layer, in addition to hiding header information. This will be explained in detail in chapter 4. The conventional mode of the C-Sec, which is used less often, has a similar header-size to the smallest of the related works that provide all required security services at the data-link layer.

Table 3.2 shows a comparison between encryption primitives used for each of the related protocols. Some of the protocols trade off security with energy efficiency by providing multiple security levels, like TinySec, MiniSec, SecureSense and FlexiSec. Many of these levels do not include all the necessary security services. Providing partial security leaves vulnerabilities from which a WSN can easily be attacked. An example of that is the lack of encryption in the TinySec-Auth mode and some of the FlexiSec modes.

Table 3.1 Header size comparison (Bytes)

Protocol	Preamble	MAC	Source address	Destination address	Length	Counter	CRC	Grp	AM	SCID	Random Number	Header Size
Zigbee	6	4	2	2	1	4	2	0	1	0	0	22
TinySec-Auth	6	4	2	2	1	2	0	0	1	0	0	18
TinySec-AE	6	4	0	2	1	0	0	0	1	0	0	14
SNEP	6	8	2	2	1	0	0	0	1	0	0	20
MiniSec-U	6	4	2	2	5-bits	3-bits	0	0	1	0	0	16
MiniSec-B	6	4	2	11-bits	5-bits	1	0	0	1	0	0	16
LLSP	6	4	2	2	1	0	0	0	1	0	0	16
SenSec	6	4	0	2	1	0	0	1	1	0	3	18
SecureSense Only Conf.	6	0	2	0	1	0	2	0	0	1	0	14
SecureSense No Semantic	6	4	2	2	1	0	0	0	0	1	0	16
SecureSense Max Sec	6	4	2	2	1	4	0	0	0	1	0	20
FlexiSec HASH	6	4	2	0	1	0	0	0	1	0	0	14
FlexiSec AUTH_REP32	6	4	2	2	1	0	0	0	1	0	0	16
FlexiSec AUTH_ENC_REPP64	6	8	2	2	1	0	0	0	1	0	0	20
WSNSec	6	4	2	2	1	2	0	0	1	0	0	18
C-Sec Conventional	**6**	**4**	**2**	**2**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**16**
C-Sec Compact	**6**	**0**	**0**	**15-bits**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**10**
TinyOS	6	0	0	2	2	0	2	1	1	0	0	13

Table 3.2 Encryption primitives comparison for related work

Protocol	MAC Algorithm	Encryption Algorithm	Implementation	Encryption Block/Key/Rounds	RAM	Code size	Overhead over TinyOS	Energy /Frame	Notes
SNEP	CBC-MAC	RC5-CTR	Software	64/128/18	220	1594	29%	-	-
TinySec	CBC-MAC	Skipjack-CBC	Software	64/80/32	728	7146	10%	1.9 *mJ*	24 byte payload
Zigbee	CBC-MAC	AES-CCM	Hardware	128/128/10	N/A	N/A	-	9.7*mJ**	Estimation based on CC2420 implementation and 30 bytes payload
MiniSec	CBC-MAC	Skipjack-CBC	Software	64/80/32	874	16 KB	8.3%	-	24 byte payload
LLSP	CBC-MAC	AES-CBC	Software	128/128/10	-	-	-	1.35*mJ**	30 byte payload
SenSec	CBC-X	Skipjack-X	Software	64/80/32	-	-	-	1.21*mJ**	
SecureSense	CBC-MAC	RC5-CBC	Software	64/128/18	-	-	25%	0.35 *mJ*	20 byte frame, Max security level
FlexiSec	SHA-1, OCB-MAC	XXTEA/AES	Software	64,128/128,128/ 32,10	600, 1.3k	11k, 25k	-	-	Max security level
WSNSec	CBC-MAC	SEA-CTR	Software	96,192/96,192/ 94,176	-	-	-	-	-
C-Sec	SHA-256	AES-CFB	Hardware	128/128/10	N/A	N/A	-6%	0.155 μJ	30 byte payload Based on SN-Sec

* Values have converted units

The lack of encryption does not provide confidentiality for messages, which facilitate message spoofing and makes it easy to launch many attacks, like attacks on routing information, selective forwarding and overwhelm attacks. Another example is lack of replay protection service in the TinySec-Auth, “Only confidentiality” mode of SecureSense and FlexiSecAUTH_ENC64 mode. This facilitates attacks based on replaying messages like desynchronization and Overwhelm Attacks. The MiniSec-U mode provides weak replay protection through a 3-bit counter. This degrades the protection against attacks based on replaying old messages.

Various encryption algorithms were used with the related protocols. Encryption algorithms have a large impact on security and efficiency. The security of encryption algorithms depends on the sophistication of the encryption operations and on the key size. 80-bit encryption algorithms used with TinySec, MiniSec, and SenSec are not secure against brute force attacks [55]. The encryption modes of operation also have impact on efficiency. For example, the CBC encryption mode does not support variable length messages. C-Sec uses AES encryption algorithm, which is a 128-bit key encryption algorithm that was selected as standard because of its high security level and efficient implementations. It also uses the CFB encryption mode, because it supports encrypting variable length messages, as well as precomputing the encryption/decryption of the next block, which saves energy and time.

Most of the protocols used CBC-MAC compute MACs, which is not secure for variable length messages [56]. FlexiSec uses SHA-1 to compute MACs. SHA-1 was reported to be broken and it is no longer recommended by the NIST after 2010 [62]. C-Sec uses SHA-256 to compute MACs. SHA-256 does not have any security weaknesses so far. The C-Sec protocol uses efficient hardware implementations of the AES and SHA-256 algorithms provided by the SN-Sec platform. These algorithms have high security levels and their implementations can run in parallel. The comparison between hardware-based implementations and software-based implementations in Table 3.2 is not apple-to-apple comparison, because hardware-based implementations are well known to be more efficient. However, the implementations of encryption primitives for C-Sec are more efficient than their hardware counterparts as will be shown in detail Section 3.3 and Chapter 5.

3.3 Selection of Hardware Implemented Encryption Primitives for WSNs.

This section provides a review of the main security primitives need to implement security services for WSN platforms. This includes three types of primitives: symmetric, asymmetric, and secure hash primitives. The review involves a comparison of algorithms proposed in the literature for each type, the selection of the most suitable primitives for WSNs based on their security, and an evaluation of their compatibility with the WSNs constraints.

3.3.1 Symmetric encryption

Many symmetric algorithms exist in the literature, the most efficient and secure ones are the AES finalists selected in the second AES conference to replace the Data Encryption Standard (DES) [63]. The security and efficiency of the AES finalists' algorithms has been intensively scrutinized by a large number of cryptanalytic experts. In the third AES conference, the Rijndael algorithm won the competition and was announced by the National Institute of Standards and Technology (NIST) as the AES algorithm [18].

However, many other algorithms were used for security protocols in WSNs. These protocols were chosen because they involve less computation, they have small key sizes, and it consumes fewer resources to implement them on WSNs platforms. As Table 3.2 shows, some of these primitives' uses 80-bit keys like the Skipjack algorithms used by TinySec and its variant Skipjack-X used by the SenSec protocols. These algorithms cannot be considered secure anymore because it is computationally affordable to break 80-bit key algorithms using brute force attack after 2012 [55].

Other algorithms like RC5, XXTEA, and SEA consume less memory resources than AES in software implementations [64]. However, AES design is more secure than these algorithms [65]. In addition, it has less number of rounds and it can be implemented more efficiently in hardware [66].

AES has three variants, a 128-bit key-size variant with 10 rounds, a 192-bit key-size variant with 12 rounds, and a 256-bit key-size variant with 14 rounds. Each round consists for four transformations performed on a 4×4 matrix called the state; byte substitution, shifting rows, mixing columns, and adding the round key. The last round of encryption/decryption does not include mix columns transformation. Figure 3.11 shows the 128-bit key variant of AES, it is the most suitable with constraint nature of WSNs.

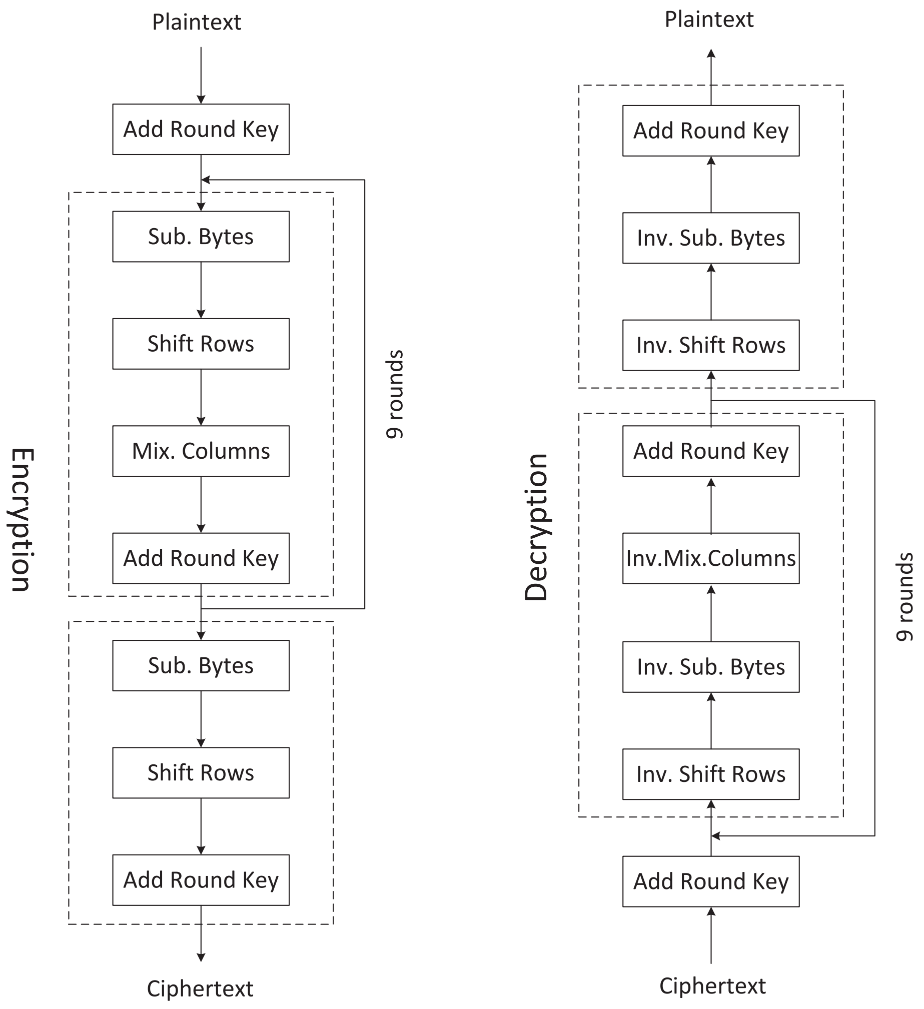


Figure 3.11 AES Algorithm

Many hardware architectures for AES were proposed in the literature. Loop-unrolled architectures are the fastest and pipelined architectures are the best in terms of high throughput [67]. However, the energy and area requirements of these architectures exceed the limitations of WSNs. The best choice for small area and energy constrained implementations are the iterated architectures.

Two main variations between iterated architectures exist in the literature. The first one is the data-path width; 8-bit, 32-bit, and 128-bit architectures were proposed. In general, higher data-path width architectures will have larger area and power requirements but smaller latencies. The authors in [68] provided energy per encryption comparisons between the three data-path width choices using implementations on UMC 0.25 micrometer 1.8v hardware. As Table 3.4 shows, although the 128-bit data path has the largest area and power usage; it has the smallest energy consumption per encryption. The 32-bit architecture has medium area and power consumption, but the worst energy consumption per encryption.

Although 128-bit architecture has the least energy consumption per encryption; the 8-bit architecture is considered to be a better choice, because it is the best in terms of area and power. As a result, it will be the best choice in terms of static energy consumption (energy consumed while the hardware is idle). The second variant between iterated architectures is the number of S-boxes used and their architectures. The 8-bit data-path design that uses two S-boxes in Table 3.4 completes one encryption in 160 cycles. Feldhofer et al. [64] implemented 8-bit data-path architecture with one S-box. Their design has an area size of 3400 gates. However, it takes 1016 clock cycles to perform one encryption, resulting in a higher energy per encryption for their design compared to the design with two S-boxes in [70]. A larger number of S-boxes are used with higher data-path widths. However, this will lead to an increase in the area and energy consumption, which is not suitable for WSNs constrained energy resources. It is shown by Mentens et.al. [66], that implementing the S-box in the $GF(((2^2)^2)^2)$ composite field rather than the $GF(2^8)$ field is more energy efficient.

Table 3.3 Different AES data-path width comparison form [68]

Data-path width	S-Boxes	Area (gates)	Power (mW)	Clock cycles	Throughput (Mbps)	Energy/Enc. (nJ)
8-bit [64]	1	3400	0.045	1016	-	-
8-bit	2	4023	1.06	160	8.1	169.6
32-bit	4	7732	3.73	54	23.7	200.5
128-bit	16	15980	11.61	11	116.4	127.7

3.3.2 Secure Hashing

Most WSN security protocols use encryption algorithms with cipher block chaining mode to produce the MACs. This method does not require implementing a separate MAC algorithm; however, it is not secure for variable-length messages [56]. Therefore, a standard hashing algorithm is selected to produce the MAC. Another advantage of using separate hash algorithm is that it can run in parallel with the encryption algorithm, which saves computation time and energy.

The two most commonly used cryptographic hash functions in the literature are MD5 and SHA-1. However, MD5 was recently broken with an attack against it used to break SSL in 2008 [72]. Some theoretical weaknesses in SHA-1 were discovered [68] [69], and two successful attacks were reported in 2005. As a result, it was decided that it will not be used by USA government agencies after 2010 [62].

New security systems will be using more advanced hash functions, such as SHA-2 family, or techniques that do not require collision resistance, such as randomized hashing [75]. NIST started a new competition to design a replacement for SHA-2 to ensure the long-term robustness of hash functions. Keccak [76] algorithm was selected in 2012. Table 3.4 shows a comparison between secure hashing algorithms, SHA-256 has the least number of iterations, smaller block size, it operates on 32-bit words, and its security is equivalent to 128-bit symmetric key encryption. However, compared to SHA-1, SHA-256 has larger number of constants, this requires larger hardware area for their implementation; it also has larger message digest i.e. 256, however, it can be truncated required size.

Table 3.4 SHA algorithms comparison

Algorithm	Block Size	Iterations	Word Size	Round Constants	Digest Size	Equivalent Security
SHA-1	512	80	32	4	160	Broken
SHA-256	512	64	32	64	256	128
SHA-384	1024	80	64	80	384	192
SHA-512	1024	80	64	80	512	256
SHA-3	1600	96	64	32	256	128

To have a more clear choice all SHA family algorithms were implemented in VHDL and synthesized for Xilinx low power Spartan 6 FPGA [77]. As Table 3.6 shows, SHA-256 has the least energy per block and running time per block results because it has only 64 iterations compared to 80 iterations for other algorithms. On the other hand, SHA-256 has 25% larger area than SHA-1 due to the large memory needed to store the algorithm constants. SHA-256 uses 72 constants compared to only nine for SHA-1. SHA-3 is showing the worst results due to the large hardware elements requirements. The details of constant usage and memory sizes used to store them are shown in Table 3.7. All time results are acceptable as WSNs have sparse and loosely coupled communication pattern.

Table 3.5 SHA family implementation results

Algorithm	Energy per Block	Frequency	Time per Block	Area	
	nJ	MHz	ns	Slices	LUTs
SHA-1	12.23	119.8	676	960	1290
SHA-256	10.76	108.7	590	1071	1891
SHA-384	17.09	105.6	918	2410	3666
SHA-512	17.46	105.5	919	2410	3794
SHA-3	21.65	403.6	2448	2874	4280

SHA-384 and SHA-512 have more than twice the area of SHA-256 because they operate on 64-bit words, and each one uses 88 constants. SHA-384 has less Look Up Tables (LUTs) count than SHA-512 because the four right most words of the final hash are truncated.

Table 3.6 Size of constants for SHA family members

Algorithm	Round const.	Initial hash const.	Total const.	Const. size	Total Size
SHA-1	4	5	9	32	288
SHA-256	64	8	72	32	2.304
SHA-384	80	8	88	64	5632
SHA-512	80	8	88	64	5632
SHA-3	96	8	104	64	6656

Although the memory consumption to store constants for SHA-256 is more than SHA-1; the results show that SHA-256 has the better energy consumption per block because it has less number of rounds. SHA-384 and SHA-512 have 60% more energy consumption per block, and more than twice the area of SHA-256. That is because they operate on 64-bit words for 80 rounds.

3.3.3 Asymmetric primitives

Many security mechanisms proposed for WSNs in the literature avoided asymmetric primitives because of their high energy and time costs, especially for software implementations. The notion of infeasibility of these primitives has been changed partially due to the development of new asymmetric algorithms that are more efficient than RSA [73]. For example, the Rabin signature algorithm [74] is very similar to RSA; its main advantage is the speed of its encryption and signature verification operations. A disadvantage is the signature size, as a single signature requires 512 bits. NTRU [80] is much faster, for both encryption and verification operations than RSA. On the other hand, it also shares the Rabin scheme disadvantage; its signature requires 1169 bits.

The SecFleck is the first recognized hardware platform that was used to provide asymmetric encryption for WSNs [19]. It is based on a commodity Trusted Platform Module (TPM) chip that extends the capability of a standard node. Although this implementation can perform 2048 bit RSA encryption in 500ms with 11mJ of energy; it has many shortcomings. Firstly, RSA has a large key size; this results in large hardware area and energy consumption for the implementation. Secondly, transmitting a large key on the wireless medium is expensive in terms of energy consumption. Thirdly, storing the keys on an EEPROM makes it easy to compromise the node by resolving the keys.

The Elliptic Curve Cryptography (ECC) [76] is extensively considered as the most suitable asymmetric cryptographic primitive for WSNs. The per-bit security for ECC is higher than other asymmetric algorithms. For example, the security of a 160-bit key for ECC is equivalent to 1024 bit key of RSA [82]. This relatively small key size results in a small hardware area, short running time, and most importantly it reduces the energy needed to transmit the key in a wireless medium. The energy cost of transmitting one bit is equivalent to the energy consumed in more than a thousand CPU cycles of a standard WSN node processor [83].

Elliptic curve cryptography is based on algebraic concepts related with elliptic curves over Galois fields. An elliptic curve can be described as a set of points that satisfy a cubic equation of a general form like in Equation 3.1, in addition to the point of infinity ∞. Elliptic curves can be defined over binary fields $GF(2^n)$ and prime fields $GF(P)$.

$$y^2 + xy = x^3 + ax^2 + b \qquad\qquad b \neq 0 \tag{3.1}$$

Given a curve defined as in Equation 3.1 in a Galois field, point multiplication can be defined as the repeated addition of a point $P = (x, y)$ on that curve for a scalar value of n. As shown in Equation 3.2. The security of the elliptic curve encryption is determined by the intractability of defining n from $Q = nP$ when values of Q and P are known, this is defined as the discrete logarithm problem.

$$nP = P + P + P + \cdots + P \tag{3.2}$$

As Figure 3.15-a shows, point addition is geometrically defined as taking two points on the curve and computing where a line through them intersects the curve. The negative of the intersection point is the result of the addition. The operation can be denoted by

$$P_1 + P_2 = P_r$$

$$(x_1, y_1) + (x_2, y_2) = (x_r, y_r) \tag{3.3}$$

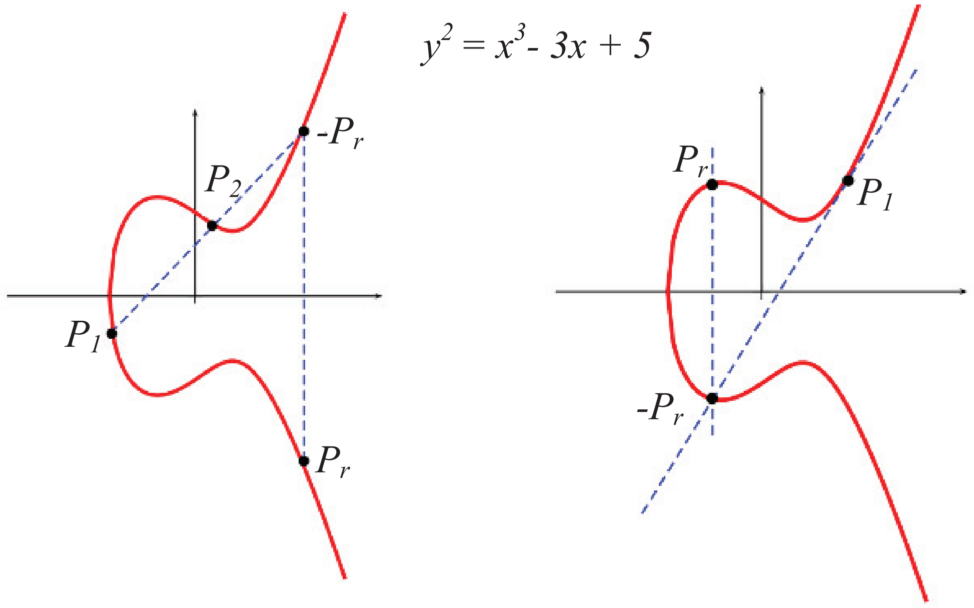


Figure 3.12 a) Elliptic curve point addition. b) Elliptic curve point doubling

This is algebraically computed by:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.4}$$

$$x_r = \lambda^2 - a - x_1 - x_2 \tag{3.5}$$

$$y_r = \lambda(x_1 - x_r) - y_2 \tag{3.6}$$

Substituting a zero value for a, where a is the multiplication factor of x^2 in the elliptic curve at Equation 3.1, reduces the complexity of the equation. This is commonly used in elliptic curve cryptography.

Figure 3.15-b shows the geometric definition of point doubling is the negative of the intersection of the tangent of a single point on the elliptic curve. The algebraic definition of point doubling is:

$$\lambda = \frac{3{x_1}^2 - 2ax_1 + b}{2y_1} \tag{3.7}$$

$$x_r = \lambda^2 - a - x_1 - x_2 \tag{3.8}$$

$$y_r = \lambda(x_1 - x_r) - y_2 \tag{3.9}$$

Adding a point to itself repeatedly as in Equation 3.2 requires $O(n)$ point additions to compute elliptic curve point multiplication. The double-and-add scalar point multiplication algorithm shown in Algorithm 3.1 provides more efficiency because it requires $O(\log n)$ operations to compute scalar point multiplication. Many variations of this algorithm exist in the literature, such as the Windowed method, and Sliding-window method which trades slower point additions for point doublings. The Non-Adjacent Form algorithm which uses the fact that point subtraction is just as easy as point addition to perform the fewer of either as compared to a sliding-window method. The Montgomery ladder algorithm [79], which has the same speed as the double-and-add approach, however, it computes the same number of point additions and doubles at each iteration. This means that it does not leak any information through timing or power.

Algorithm 3.1 Double-and-add scalar point multiplication

```
$\boldsymbol{Input}: P: EC\ Point, Scalar\ n > 0,\ n: n_{l-1}, \cdots, n_1, n_0.$
$\boldsymbol{Output}: Q = nP$

    $for\ i = l - 2\ down\ to\ 0$
        $Q = 2Q$
        $if\ (n_i = 1)$
            $Q = Q + P$
    $return\ Q$
```

The main ECC primitive operation is the scalar point multiplication. It consists of a series of point additions and doublings, which are of a series of modular: addition, multiplication, squaring, and inversion operations; according to a specific coordinate system. Many projective coordinate systems were proposed in the literature to avoid the high cost of modular inversion that is repeated in each point addition and doubling in the affine coordinates. Many new curve representations associated with projective coordinates systems are being proposed. Montgomery coordinates [79] (also known as XZ projective coordinates) is the most efficient so far [80]. Table 3.7 shows some common coordinate systems and their costs in terms of number of inversions (I), multiplications (M), squaring (S).

Table 3.7: Elliptic curve coordinates systems comparison

Coordinate system	*EC Doubling*	*EC Addition*
Affine	1I + 2M + 2S	1I + 2M + 1S
Standard Projective	7M + 3S	12M + 2S
Jacobian Projective	4M + 4S	12M + 4S
Lpez-Dahab projective	2M + 4S	4M+2S
XZ coordinate projective	2M + 2S	3M + 2S

Redundant interleaved modular multiplications [81], Barrett multiplication [82], and Montgomery modular multiplication [83] with carry save addition are the most efficient modular multiplication algorithms proposed. Redundant interleaved multipliers are reported to have some advantage over Montgomery's in area and speed. However, using one carry save adder with Montgomery algorithm makes it similar to Redundant Interleaved multipliers in area with small advantage over higher precision. However, Montgomery modular multiplication calculates $X.Y.2^{-n} mod\ M$ instead of $X.Y\ mod\ M$ more operations are required to convert it back from the Montgomery domain [84]. Barrett multiplication and Montgomery multiplication have similar area and timing results [85].

Choosing the domain parameters of the elliptic curve is another matter of security and efficiency. The NIST and Securities Exchanges Guarantee Corporation (SEGC) proposed curves and domain parameters for ECC [82] [91]. Table 3.8 shows the curve field sizes, their symmetric equivalent strength. The Certicom ECC challenge [92] classified the curves in to two levels. Level I curves are considered feasible to break, i.e. they could be broken within a few months using brute force attack. In July 2009, a 112-bit prime field ECC case was broken using a cluster of over 200 PlayStation 3 game consoles within 3.5 months [93]. Before that, a 109-bit key binary field case was broken in April 2004 using 260 computers for 17 months [89]. A current project in progress by Certicom is aiming at breaking the 130 bit extension field using a wide range of different hardware and it is expected to be broken soon [95].

Choosing a curve within a field size that is larger than 193 bits is recommended for both prime and binary fields. Fields in the 163-bit extension fields security is marginal because they are classified as equivalent to 80 bit symmetric encryption keys, which are not considered secure after 2012 [55]. SEGC removed some of these curves in the latest revision of their standards [91].

Choosing binary fields over prime fields is more attractive for energy efficient hardware implementations. Firstly, bit additions within binary fields are done modulo 2; hence they are represented in hardware by simple XOR gates. In addition, bit multiplications in the binary fields are represented in hardware by AND gates. Finally, the one is its own inverse $1 = 1^{-1}$ in binary fields. Hence, implementing the elliptic curve processor in binary fields simplifies computations and reduces both, the hardware area and energy consumption. In addition, binary fields offer more options in terms of bases, irreducible polynomials and fields [92].

Table 3.8: Security status of curves in $GF(P)$ and $GF(2^n)$.

Prime Fields	*GF(109)*	*GF(131)*	*GF(163)*	*GF(191)*	*GF(239)*	*GF(359)*
Strength	56	64	80	96	112	128
Level	I	I	II	II	II	II
Broken	In 2009	In progress	No	No	No	No
Binary Fields	*GF(109)*	*GF(131)*	*GF(163)*	*GF(191)*	*GF(238)*	*GF(359)*
Strength	47	64	80	96	112	128
Level	I	I	II	II	II	II
Broken	In 2009	In progress	No	No	No	No

Some specialized ECC hardware implementations for WSNs have been proposed recently. Table 3.9 shows the most well-known elliptic curve processors implementations in hardware for WSNs. Comparing based on power consumption, frequency, and time results does not give a true indication of the quality of the algorithm used to compute the elliptic curve point multiplication. The variation in field size and process technology has large effect on these factors.

Instead, the related works are compared based on the number of mathematical operations needed to perform point addition and point doubling, and the total number of operations needed to perform elliptic curve point multiplication. The processors in [98-101] make use of affine coordinate system which includes inversion which is considered very expensive compared to modular multiplication. The proposed elliptic curve processor has the least number of operations for computing point addition and doubling. It also exploits the use parallelism between operations with no dependency to reduce the number of operations to compute elliptic curve point multiplication.

Table 3.9 Elliptic curve hardware implementations for WSNs

	Point addition	Point doubling	Point Multiplication	Coordinate system	Galois Field size	Gate equivalence	Power (mW)	Technology	Frequency (MHz)	Time (ms)
[93]	3M + I	3M + I	n(4.5M + 2I)	Affine	167	112K	150	0.5 µm	20	4.4
[94]	3M + I	3M + I	n(4.5M + 2I)	Affine	101	18.7k	0.4	0.13µm	0.5	410
[100]	I+M+S	I+M+S	n(2M+2I+2S)	Affine	193	18k	-	0. 35µm	13.56	38.8
[96]	3M + I	3M + I	n(6M+I)	Affine	191	23k	-	0.13µm	68.5	18
[97]	2M + 4S	5M+3S	I+n(7M+4S)	Common Z Projective	163	12k	0.037	0.13 µm	1.13	244
[98]	6M + 1S	3M + 5S	I+n(9M+6S)	Modified Lopez-Dahab	163	5k	-	-	1	500
[104]	4M+S	2M+5S	I+n(6M+6S)	Modified Lopez-Dahab	163	FPGA	-	0.13µm	0.5	115
[20]	-	-	-	-	283	98k	854	-	25	0.75
Proposed	3M+2S	2M+2S	I+n(5M)	XZ coordinate	193	FPGA	27	-	58	3.2

3.4 CONCLUDING REMARKS

This chapter presented the security and energy efficiency factors related to the design of link layer encryption protocols. The design of the proposed C-Sec protocol considers these factors to avoiding security weaknesses of other protocols and to provide more energy efficiency. It also provided guidelines for encryption primitives' selection and implementations for both the C-Sec and the SN-Sec.

Chapter 3 started by reviewing the TinyOS characteristics and frame format. TinyOS is an operating system that is widely accepted for WSNs, and most of the related encryption protocols are designed based on it. An extensive review for the most recognized link-layer encryption protocols designed to implement the basic security services for WSNs was also provided. The review included the encryption primitives used by each of these protocols, their packet formats, modes of operation, security services provided, and the mechanisms used by each protocol to minimize the size of header fields added to implement security services.

A review of encryption primitives implemented in hardware for use of WSNs was presented in section 3.3. These primitives include symmetric, asymmetric and hashing primitives. Based on this review, recommendations for the selection of encryption primitives for the SN-Sec platform were provided. For the symmetric encryption, we found that AES has the highest level of security. It is also very efficient for hardware implementations, with low energy consumption, latency, and hardware area. Hence, it is the best choice for hardware implementations for WSNs. After reviewing the low power hardware designs of AES in the literature, we found that the 8-bit data-path design with two S-boxes to be the most efficient data-path design [70]. We also found that the most efficient design for the S-box is in the $GF(((2^2)^2)^2)$ composite field [66]. Combining the 8-bit data-path design in [70] with the S-box design in [66] will make the implementation of the AES algorithm more efficient.

We implemented all SHA algorithm family members in VHDL and a comparison between synthesis results of their implementations was performed. We found that the SHA-256 has the least per block energy consumption and latency. It also has the least number of rounds and it operates on 32-bit data-path compared to other SHA algorithms. The implementations and comparisons performed prove that SHA-256 is the best choice of secure hashing algorithms for WSNs.

Elliptic curve encryption has the highest per-bit security among asymmetric encryption algorithms. It has a small key size and it operates on a smaller number of digits. Hence, it has a smaller hardware area, a shorter running time, and most importantly less energy is needed to transmit the ciphertext and other elliptic curve parameters over the wireless medium. An overview of the elliptic curve operations, parameters, coordinate systems, and security status of the fields was presented. And a comparison of elliptic curve processors proposed for WSNs in the literature was performed and recommendations for the proposed elliptic curve design were given.

CHAPTER 4 THE COMPACT SECURITY PROTOCOL

Communication energy is the main source of energy consumption for WSNs. As explained in section 3.2, all encryption protocols proposed for WSNs in the literature aim to reduce the security related communication overhead in different ways to save communication energy.

The saving in communication energy for the C-Sec protocol is a result of the innovative idea of excluding the requirement for explicitly transmitting all header fields related to security most of the time, while keeping all related security services. Such fields include the freshness counter, the source address, and the Message Authentication Code (MAC). As shown in Table 3.1, the header size of the compact mode of the C-Sec protocol is at least 6-bytes less than the header size of other encryption protocols that implement all the basic security services. It is even 3-bytes less than the header size of the TinyOS packet, which does not implement any security service. Using the compact mode most of the time will result in reducing the communication energy to less than the energy consumed if no encryption protocol is used at all.

The C-Sec protocol provides all the basic security services that are provided by other security protocols in the literature, such as data authentication, integrity, confidentiality, semantic security, and replay protection. However, it adds a new unique security feature of hiding the packet header, making it more difficult to eavesdrop on the flow of wireless communication between nodes and minimizes the cost of defending against replay attacks. This feature does not exist in any previous protocol for WSNs in the literature.

In this chapter, a detailed description of the C-Sec protocol is provided and the security services it provides and the underlying cryptographic algorithms it uses are analyzed. To prove the efficiency of the C-Sec, performance evaluations for energy, queuing delay, and error probability are presented based on simulations and mathematical models.

4.1 PROTOCOL DESCRIPTION

The C-Sec protocol operates in one of two modes, conventional or compact. Like all other encryption protocols, the conventional mode of the C-Sec adds security related fields, such as the source address and the MAC, to the packet header and trailer. The conventional mode maintains the freshness counters on both sides of communication without transmitting them, as in SNEP. It uses the most significant bit of the destination address to transmit security related information. However, it uses only one bit, this will not dramatically affect the address space as in MiniSec-B. Like most of the related protocols, the conventional mode of C-Sec does not use the Grp field because its functionality is implemented in the keying mechanism. It also does not use the CRC field because its functionality is implicitly implemented with the MAC field. As shown in Figure 4.1, the header of the C-Sec conventional mode frame consists of the basic fields only, such as the mode bit M, which is used to differentiate between the conventional and the compact mode of the C-Sec, a 15-bit Destination address $Dest$, one-byte Active Message AM, and the Length Len fields. The trailer consists of a four-byte MAC field.

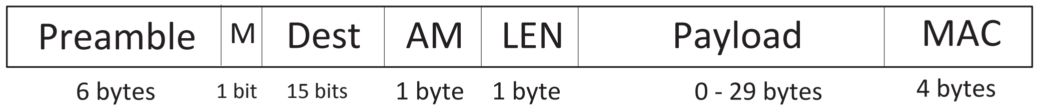


Figure 4.1 The C-Sec conventional mode frame

The conventional mode can be described with Equations 4.1-4.3. Where H_i is the header of the packet, T_i is the trailer of the packet, M_i is the payload, C_i is the freshness counter, K_{Auth} is the authentication key, and K_{Encr} is the encryption key:

$$T_i = MAC\left(K_{Auth}, AM, Len, E_{K_{Encr}}(M_i), C_i\right) \tag{4.1}$$

$$H_i = M{:}\, Dest : AM : \, Len \tag{4.2}$$

$$A \rightarrow B \quad H_i : \, E_{K_{Encr}}(\, M_i) : \, T_i \tag{4.3}$$

To start the C-Sec protocol, node authentication-and encryption keys have to be exchanged in advance. Communicating parties are assumed to have already exchanged those keys using any key management protocol for WSNs. [105] provides a comprehensive survey of key management protocols for WSNs.

The compact mode of the C-Sec has smaller frame size, because it does not explicitly transmit the MAC and the destination address fields, but merges them with other header fields. This will result in hiding the header fields and it does not affect the basic security services.

The C-Sec protocol is initiated in the conventional mode. On both sides of communication, the sender and receiver should store the MAC of the first packet communicated in the conventional mode. The MAC is computed by running a one-way hashing algorithm, with the encrypted message $E_{K_{Encr}}(M_i)$, authentication key K_{Auth}, and freshness counter C_i as inputs. On the sender side, instead of adding the MAC to the message as a standalone field (the practice of all other protocols); the most significant 31-bits of the MAC of the previous encrypted message $E_{K_{Encr}}(M_{i-1})$ is XORed with the header of the current message H_i to produce a 31-bit Masked Header MH_i for the current message M_i. Figure 4.2 shows the frame format of the compact mode. Equations 4.4 and 4.5 illustrate its behavior.

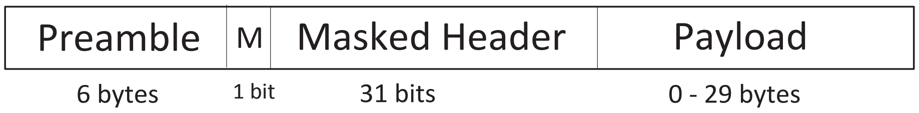


Figure 4.2 C-Sec compact mode frame

$$MH_i = H_i \oplus MAC\left(K_{Auth}, AM, LEN, E_{K_{Encr}}(M_{i-1}), C_{i-1}\right) \tag{4.4}$$

$$A \rightarrow B \qquad MH_i : \; E_{K_{Encr}}(M_i) \tag{4.5}$$

As Figure 4.3 illustrates, the conventional mode is denoted with a zero value of the M bit and the compact mode with the value of one. The compact mode of C-Sec is automatically initiated starting from the second packet on the sender side. All following packets are sent in the compact mode as long as the time between them and their previous packets remains less than a specific time limit called the authentication-timer. If the authentication-timer expires for any node, the next packet will be sent to it in the conventional mode. The authentication-timer aims to limit the amount of delay overhead, storage and computations introduced by C-Sec protocol. The conventional mode can also be enforced on demand by the application layer in the case of high bit error rates or QoS time constraints.

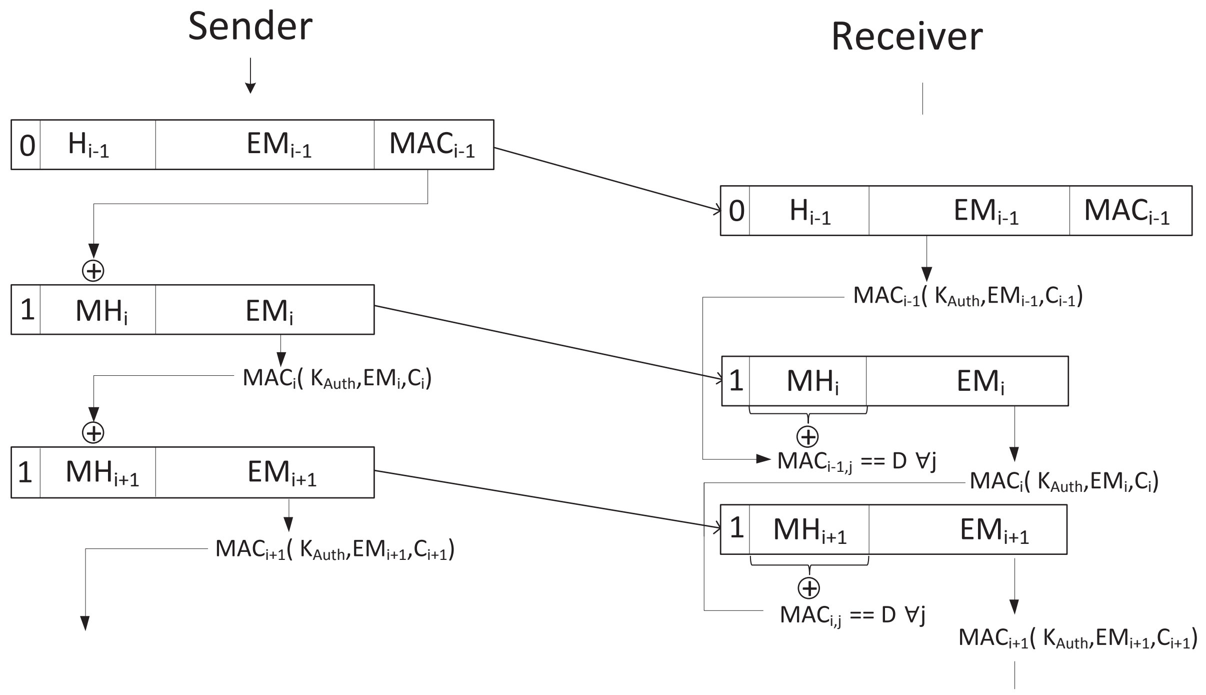

Sender
Receiver
0 | H_{i-1} | EM_{i-1} | MAC_{i-1}
1 | MH_i | EM_i
1 | MH_{i+1} | EM_{i+1}
$MAC_i(K_{Auth}, EM_i, C_i)$
$MAC_{i+1}(K_{Auth}, EM_{i+1}, C_{i+1})$
$MAC_{i-1}(K_{Auth}, EM_{i-1}, C_{i-1})$
$MAC_{i-1,j} == D\ \forall j$
$MAC_{i,j} == D\ \forall j$

Figure 4.3 The behavior of the C-Sec protocol

For each peer-node being communicated with, an entry should be created to store the information needed by C-Sec in the cache. This information includes its address, the freshness counter, a time stamp of the last received packet, which can be used to compute the authentication timer, and the MACs for the current and last received packets. Table 4.1 shows the node entry in cache.

Table 4.1 Node entry in cache

Peer node address	MAC_{i-1}	MAC_i	Freshness counter	Time stamp

The flow diagram in Figure 4.4 shows the behavior of C-Sec protocol on the receiver side. Once a packet is received at the destination, the mode of the packet is specified based on the mode bit M. If the packet is conventional, the sender address is looked up in the cache. If a match for sender j is found in the cache, the MAC is verified and stored and the packet is accepted. If the sender address is not found in the cache the authentication algorithm must be run to decide whether the packet is authentic. If so, the packet is accepted, a new entry is added for it in the cache, and the authentication timer is set for sender j. Otherwise, the packet is dropped.

If the mode is compact, the part of the masked header MH_i that maps to the destination address is XORed with the corresponding part of the pre-computed $MACs$ of the last received packets from all senders in the cache. If the result of any of the XOR operation produces the receiver's address D, the packet is held and attributed to the sender j associated with that MAC. The receiver will continue receiving the packet, set the authentication timer, decrypt the payload using the session key K_{Encr}, and update the value of the freshness counter C_i. The body of the packet will be pending for authentication when the next packet header arrives. If the authentication timer expires before the next packet from that sender arrives, the packet i is dropped and the authentication data associated with it is deleted from the cache. When the next packet from sender j arrives, the algorithm is repeated.

If none of the XOR operations produces the destination address D, the packet will be "early dropped" for one of three reasons: Firstly, it could be addressed to another node. Secondly, it could have a transmission bit error. Thirdly, it could be a malicious packet injected by an attacker. The reason of dropping the packet can be estimated by the receiving node based on many factors, like the control packets i.e. CS, RTS, and CTS, sent by other nodes, the noise measure by the transceiver and the threat measures evaluations at the application layer. Actions can be taken by the application layer in response to the reason of dropping the packet.

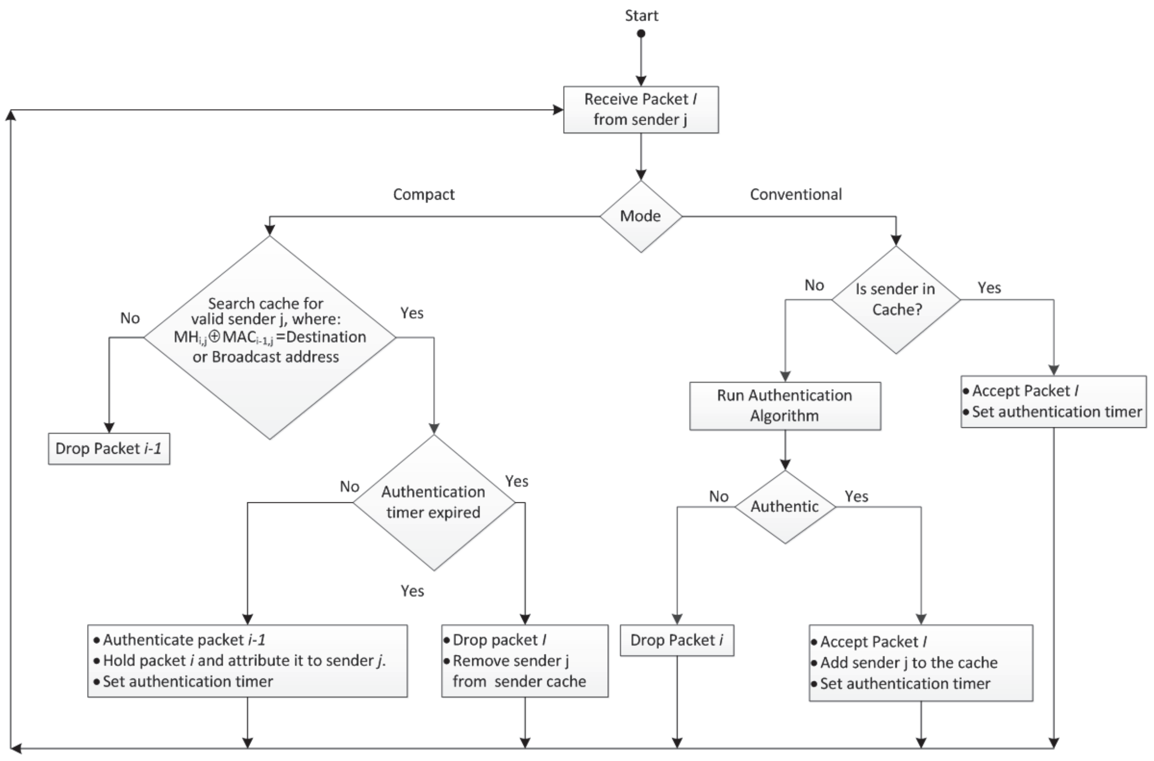

Start
Receive Packet I from sender j
Mode
Compact
Conventional
Search cache for valid sender j, where: $MH_{i,j} \oplus MAC_{i-1,j}$ =Destination or Broadcast address
No
Yes
Drop Packet i-1
Authentication timer expired
Authenticate packet i-1
Hold packet i and attribute it to sender j.
Set authentication timer
Drop packet I
Remove sender j from sender cache
Is sender in Cache?
Run Authentication Algorithm
Accept Packet I
Set authentication timer
Authentic
Drop Packet i
Accept Packet I
Add sender j to the cache
Set authentication timer

Figure 4.4 The behavior of the C-Sec on the receiver side

To prevent active attacks in the compact mode, the body of the received packet will be held back in order to be authenticated when the next packet header arrives. In cases of high priority messages or quality of service constraints, the sender can flexibly switch back and forth between the conventional and compact modes in order to prevent waiting for the next packet. If the next packet is not expected to be ready within an acceptable period, the authentication data can be sent with a standalone packet after a timeout period.

In early overhearing avoidance mechanisms [106], if the packet is not addressed to a receiver, it instantly stops receiving and moves the transceiver to a low power state before the whole packet is received. This can be decided when the receiver starts receiving the destination address and compares it to its own address. To gain the highest advantages of this mechanism, MAC computation and lookup should be completed within a limited period (i.e. the time between the end of receiving packet i and the end of receiving the preamble and the masked header of the next packet $i + 1$). Meeting this time constraint requires implementing the MAC algorithm in hardware. Hardware implementations are much faster and have much lower energy consumption than their software counterparts up to a factor of 10^{-3} [24]. For example running a software implementation of Cipher-based Message Authentication Code (CMAC) algorithm on TelosB platform for a 24 bit data packet takes 2.5 ms and consumes 13.24 μJ [107], whereas running the hardware based MAC scheme proposed in SN-Sec [24] takes 1.1 μs and consume 30.9 nJ.

Usually, nodes closer to the sink handle more traffic from the network than other nodes and hence consume more energy and die first, resulting in a disconnected network with potentially uncovered areas [108]. Our protocol helps to extend the lifetime of these nodes. The amount and frequency of the traffic they handle allows them to use the compact mode more often than other nodes, and hence save more energy and extend the lifetime of the WSN.

Masking the header of the message with the MAC makes it confidential; as a result, it will be more difficult to trace the flow of wireless communication. To get the header information, an adversary needs to get the authentication key K_{Auth} to re-compute the MAC_{i-1} and then XOR it with the masked header MH_i to resolve the original header H_i. This feature does not exist in any of the previously proposed protocols for WSNs in the literature**.**

The C-Sec compact mode is built based on - and has the same level of reliability as - stop and wait Automatic Repeat request (ARQ) strategy. Stop and wait is the most commonly used ARQ strategy for WSNs [49] [40]. In this strategy, the sender waits for the acknowledgment of the current packet before it sends the next one. Other ARQ strategies such as window-based ARQ cannot be used in the C-Sec compact mode because in the case of out of order messages, the header of the packet (i.e. destination address) cannot be identified unless all pervious messages in the window have arrived. This requires storing all traffic in the wireless medium, including traffic destined to other nodes, and performing an exhaustive search for the correct message, which is not efficient.

4.2 Security Analysis

This section analyzes the security services C-Sec protocol provides based on the underlying cryptographic primitives it uses and compares these services and primitives with related protocols in the literature. The Security services include data confidentiality and semantic security, data authentication and integrity, and data freshness and replay protection.

4.2.1 Data Confidentiality and Semantic Security

Confidentiality is achieved by encrypting the data with a secret key that is only known by the intended receivers. Semantic security ensures that the adversary cannot learn anything about the plaintext from the ciphertext. This is usually achieved by using block cipher modes of operation that support semantic security.

AES encryption algorithm is chosen for C-Sec. AES was selected as an NIST standard algorithm because it has proven security and efficient implementation in hardware and software [109]. However, hardware implementations of AES are more efficient in energy and latency; and relatively more secure than software implementations [24]. Table 4.2 shows the encryption algorithms used by related encryption protocols for WSNs.

Encryption protocols other than the AES were used because of their small key size, like the use of Skipjack algorithms for TinySec and its variant Skipjack-X used by the SenSec protocol. These algorithms cannot be considered secure anymore because it is computationally affordable to break 80-bit key algorithms using brute force attack after 2012 [47]. Other algorithms were selected by because of their relatively smaller code size and efficient software implementations, like the use of RC5 for SNEP and SecureSense, and the use of SEA for WSNSec. However, AES design is more secure than these algorithms [81]. In addition, it has less number of rounds and it can be implemented more efficiently in hardware [82].

Table 4.2 Encryption and MAC algorithms for related work

Protocol	MAC Algorithm	Encryption Algorithm	Implementation	Encryption Block/Key/Rounds	RAM	Code size
SNEP	CBC-MAC	RC5-CTR	Software	64/128/18	220	1594
TinySec	CBC-MAC	Skipjack-CBC	Software	64/80/32	728	7146
Zigbee	CBC-MAC	AES-CCM	Hardware	128/128/10	N/A	N/A
MiniSec	CBC-MAC	Skipjack-CBC	Software	64/80/32	874	16 KB
LLSP	CBC-MAC	AES-CBC	Software	128/128/10	-	-
SenSec	CBC-X	Skipjack-X	Software	64/80/32	-	-
SecureSense	CBC-MAC	RC5-CBC	Software	64/128/18	-	-
FlexiSec	SHA-1, OCB-MAC	XXTEA/AES	Software	64,128/128,128/ 32,10	600, 1.3k	11k, 25k
WSNSec	CBC-MAC	SEA-CTR	Software	96,192/96,192/ 94,176	-	-
C-Sec	SHA-256	AES-CFB	Hardware	128/128/10	N/A	N/A

C-Sec provides semantic security by running the AES algorithm using Cipher Feed Back (CFB) mode. This mode of operation has two advantages. Firstly, the message does not need to be padded to a multiple of the cipher block size and can have a variable size [40], which will save communication energy. The second advantage is the ability to pre-compute the output of the AES algorithm. As Figure 4.5 shows, the encryption of the next block will only require an XOR operation of the precomputed output T_i with the message M_i without running the AES algorithm. The same procedure applies to the decryption process. The initial vector used to initialize the CFB mode is defined as:

$$IV = MAC(K_{Encr}, K_{Auth}, C_0) \tag{4.6}$$

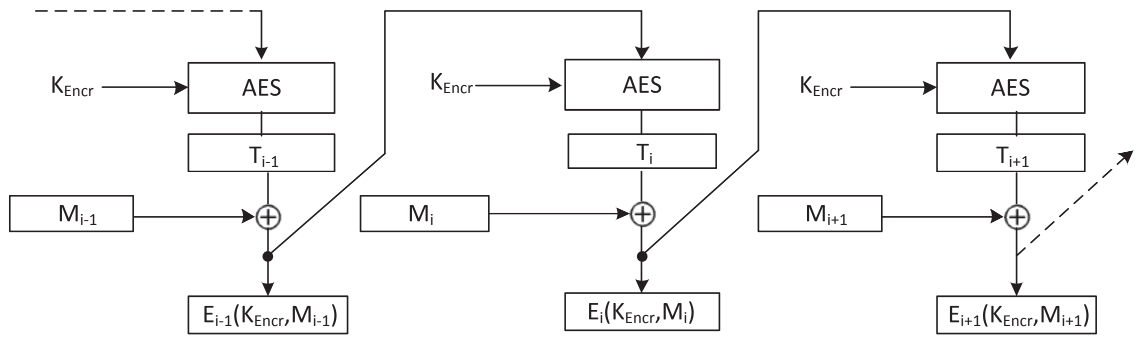


Figure 4.5 CFB encryption mode.

4.2.2 Data Authentication and Integrity

Data integrity helps the receiver to ensure that the received data is not altered by an adversary during transmission. Data authentication allows verifying that the data is sent by the claimed sender. In C-Sec, these properties are achieved by computing a message authentication code using the secret authentication key, K_{Auth} that is only known by the sender and the receiver. When the receiver verifies the correctness of the MAC of the received message it ensures that the massage was sent by the claimed sender who has the authentication key K_{Auth}, and it was not altered during transmission.

As shown in Table 4.2, most of the protocols use an encryption algorithm with cipher block chaining CBC-MAC to produce the MAC. This method does not require implementing a separate MAC algorithm. However, it is not secure for variable-length messages [56]. Thus, a standard hashing algorithm is chosen to produce the MAC. Another advantage is that the hashing algorithm can run in parallel with the encryption algorithm. This helps to meet the timing constraint of the compact mode. The MAC is computed using SHA-256 algorithm for two reasons. First, both SHA-1 and MD5 have known weaknesses and were reported to be broken [110-112]. Second, it has higher implementation efficiency compared to other secure hashing algorithms [113]. To produce the MAC, SHA-256 is run two times as stated in the Equation 4.7

$$MAC = SHA\left((K_{Auth} \oplus opad)||SHA\left((K_{Auth} \oplus ipad)||M\right)\right) \tag{4.7}$$

Where $ipad$ and $opad$ are paddings used to fit the key to the hashing algorithm key block and M is the message to be authenticated.

4.2.3 Data Freshness and Replay Protection

Data freshness is used to prevent the adversary from playing old messages. This is achieved by ensuring that the sent data is recent through maintenance of a freshness counter on both sides of the communication. This counter is used in the computation of the MAC of the message as shown in Equation 4.1.

In the case of traditional protocols, if a replayed packet is sent to a node, the node will not discover that the packet is not legitimate until it completely receives the packet, run the authentication algorithm with the freshness counter, and then compare the resulted MAC with the MAC transmitted with the packet. However, if an old packet is being replayed in the compact mode of C-Sec, the computed destination address will not be correct because the counter is wrong and the packet can be dropped without completely receiving it and without the need to run the authentication algorithm to evaluate its MAC. The cost of replay attacks is reduced by the resulting energy savings.

The C-Sec can be used as an attack detection technique. An adversary cannot get any information from a packet in the compact mode because the header is hidden and the payload is encrypted. It has to jam the network to create timeouts in the authentication timer to force the node to move to the conventional mode to at least get the packet headers. The percentage of conventional packets and the pattern of forcing the traffic to the conventional mode can give a good indication of attacks on the node.

4.2.4 Other Security Issues

C-Sec is a link layer protocol; its security is dependent on protocols and services in other layers, e.g. key management protocols. C-Sec does not explicitly deal with node compromise or physical tampering, and it does not address information leakage through covert channels. However, certain hardware specific measures can be used to prevent physical tampering, like the use of glue logic design to make reverse engineering much harder and the use of multiple metal layers to block direct access to the chip [106].

4.3 Performance Evaluation

This section presents simulation and analytical models to evaluate the C-Sec protocol. The C-Sec protocol is compared to other related protocols in the literature and with TinyOS as a baseline. The analysis demonstrates that the C-Sec improves energy efficiency over other protocols for different payload sizes and traffic densities. The additional delay overhead introduced by C-Sec and its effect on the end-to-end delay are examined, in addition to the effect of C-Sec on post-decryption error probability and packet loss.

The simulation model used is based on the Castalia [22], a WSN simulator that incorporates a realistic wireless channel, radio models, and node behavior, especially relating to the radio access. The Castalia simulator is based on OMNeT++ [115], an event driven simulation platform that is considered as a standard tool to study protocols for wired and wireless networks.

The simulation is done for a multi-hop single sink data-gathering network of 30 randomly deployed nodes in an area of 70x70 meters. The T-MAC is used as the underlying MAC protocol. The value of the authentication timer is carefully chosen to cover more than two T-MAC cycle times to allow reasonable time for contention on the wireless medium and to limit the amount of processing and the size of authentication data stored at the sender and receiver nodes. Table 4.3 shows the simulation parameters. All other parameters are set to their default values in the Castalia simulator.

Table 4.3 Simulation Parameters

Parameter	***Value***
Area Size	70 x 70
Number of nodes	30
Link layer protocol	T-MAC
Transceiver	CC2420
Payload size	30 bytes
Physical layer overhead	6 bytes
Sensitivity	-95 dBm
Noise Floor	-100 dBm
Event interarrival time	5 Sec
Packets generated by event	5
T-MAC frame time	610ms
Authentication timer	1750ms

4.3.1 Energy Analysis

As explained in Section 4.1, nodes executing the C-Sec protocol start communication in the conventional mode initially, and then it switches to the compact mode. During transmission, if the authentication timer expires, the communication goes back to the conventional mode. To evaluate the energy savings by the C-Sec protocol the total number of packets generated in the compact mode and the conventional mode are observed. The simulation experiment is done for the first of 120 seconds of the network lifetime, and repeated for on thousand times. The results with 95% confidence for this experiment are illustrated in Figure 4.6 highlighting the variation of compact and conventional mode packets with respect to time. At the beginning, it can be observed that the conventional mode packets are approximately 42 percent of the total. However, a rapid decrease is noted. Similarly, the percentage of compact packets increases very rapidly and reaches approximately 94% of the total after 80 seconds.

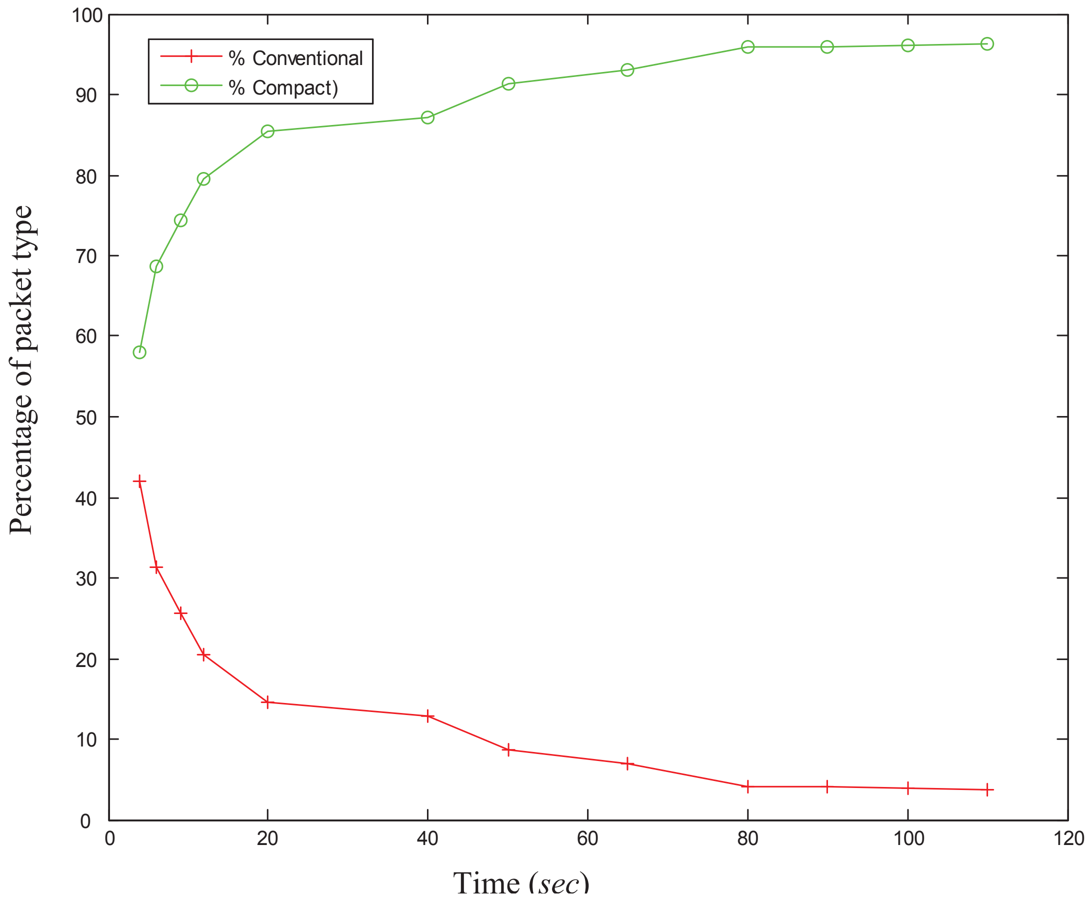


Figure 4.6 C-Sec packet breakdown

More simulations for longer time show that the percentage of compact packets is stable at around 96%. The remaining 4% of conventional packets result from timeouts of the authentication timers and due to bit errors. It can be clearly observed that a large number of compact packets contribute to significant energy savings because of their smaller size compared to conventional packets. As a result, less energy is required to transmit and receive them.

To compare the communication energy consumption of the C-Sec and the other related protocols, the simulation was run for 1000 seconds and repeated with various packet payload sizes and the average energy consumed by a transceiver is computed with a confidence level of 95%. As Figure 4.7 shows, the energy consumption increases with the increase in the payload size. It is also observed that the energy consumption varies among the protocols based on the amount of increase of the header size each protocol adds to the packet. The C-Sec consumes less energy than the TinyOS; this means that the C-Sec manages to provide all the required security services while saving on energy consumption.

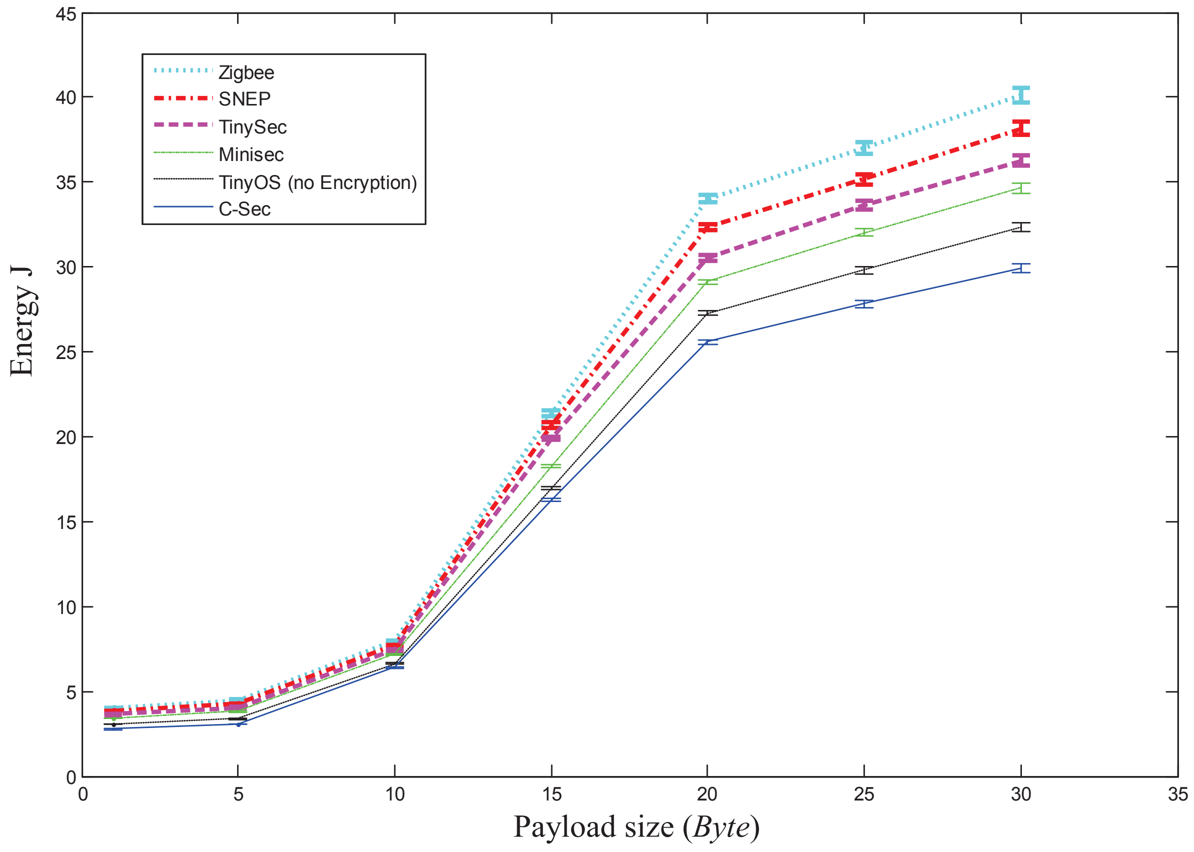


Figure 4.7 Communication energy vs. payload size

It can also be inferred from the results in Figure 4.7 that the C-Sec saves more energy than all other protocols with smaller payload sizes, which are more common in WSNs. For example the energy saving for the maximum payload size of 30 bytes is 11.7% compared to MiniSec protocol. However, for a payload size of 5 bytes the energy savings increases to 19.1%.

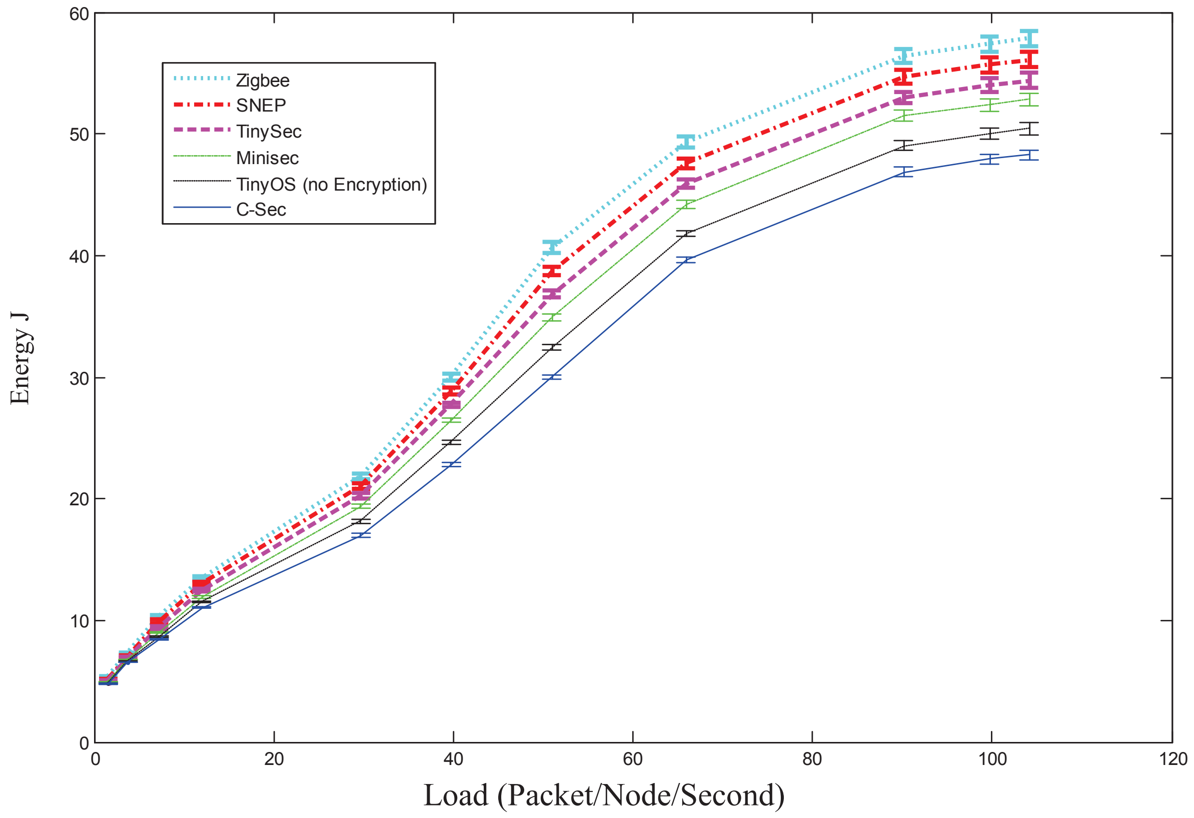


Figure 4.8 Communication energy vs. load

The energy consumption is also evaluated using different packet loads i.e. packets received per second, a fixed payload size of 30 bytes, and 95% confidence. As shown in Figure 4.8, the increase in the energy consumption depends on the amount of additional header size overhead each protocol adds to the packet. The C-Sec protocol uses less energy consumption than other protocols. The energy saving increases with higher packet loads as a result of the increased percent of compact packets, the energy savings reaches up to 11.7% at 50 packets per second compared to MiniSec, but the gap decreases again to 9.5% when the network is more saturated.

4.3.2 Computational Overhead

The computational overhead of the compact mode of the C-Sec is defined as the number of CPU cycles needed to unmask the received packet or to decide to drop it. This is a new overhead added by the C-Sec and does not exist in other related works. Checking if the received packet is destined to the receiver node involves comparing its masked header with the MACs of the last received packet from each sender in the receiving node cache.

Figure 4-9 shows the cache search algorithm used to determine if a received packet in the compact mode is accepted or rejected. Deciding if the packet is to be accepted or rejected at any iteration of the algorithm takes a maximum of four clock cycles, assuming that each condition of the algorithm can be checked with one clock cycle i.e. a 32-bit processor.

```
1.   For $j = 1; j \le$ Cache size; $j{+}{+}$
2.       If ( Auth. timer of $\mathrm{MAC_j}$ is expired )
3.           Early drop the packet $i$
4.           Delete $\mathrm{MAC_j}$ from receiver cache
5.           Break
6.       Else
7.           Temp $\Leftarrow \mathrm{MH}_i$ xor $\mathrm{MAC_j}$
8.           If ( Temp = Destination Address )
9.               Accept the packet $i$ as a unicast packet
10.              Break
11.          Else if ( Temp = Broadcast Address )
12.              Accept the packet $i$ as a broadcast packet
13.              Break
14.          End if
15.      End if
16.  End for
```

Figure 4.9. Cache search algorithm in the receiving node cache

The authentication timer is evaluated in the first clock cycle. If the authentication timer is expired, the packet is dropped and the MAC being tested is deleted from the receiver cache. If it did not expire, the result of XORing the masked header of the received packet with the MAC being tested is computed in the second clock cycle. The third clock cycle verifies if the result of the previous step is all ones, which means that the received packet is a broadcast packet. If not, the fourth clock cycle verifies if the result is the destination address of the receiving node, which means that the packet is a unicast packet destined to the receiving node. Otherwise, the packet will be dropped. These steps are repeated for each MAC in the receiving node cache.

To evaluate the computational complexity of the C-Sec protocol, simulations have been conducted with variable node densities (number of nodes ranging from 15 nodes to 1200 in an area of 70 x70 m^2). The packet size is set to one byte and the number of events per second is set to one thousand in order to guarantee generating the maximum possible packet load. The energy needed to run the cache search algorithm in Figure 4.9 was computed by counting the clock cycles needed to run the algorithm, assuming that one instruction consumes 1 nJ.

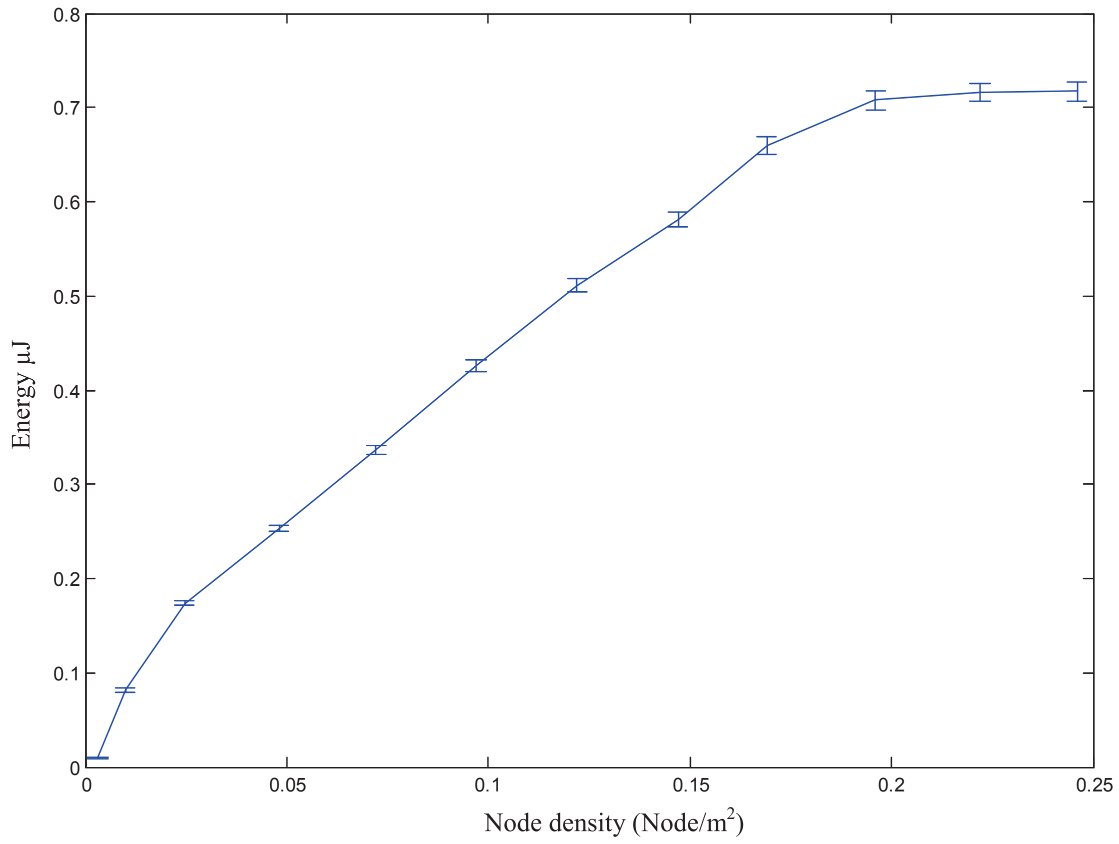


Figure 4.10. Computational overhead of the compact mode of C-Sec

As Figure 4.10 shows, the energy consumption to evaluate one received packet increases as the node density increases. However, beyond the density of 0.2 nodes per square meter it becomes stable at around 0.7 μJ because the maximum number of packets with various destination addresses that can be sent within the authentication timer period is reached at that point. Packets delivered outside the authentication timer window are sent in the conventional mode. Considering that the energy needed to transmit one bit in the wireless medium is equivalent to more than one thousand clock cycles for most of the WSN nodes [3] the energy needed to compute that is small compared to the communication energy gain of the C-Sec.

4.3.3 Delay and Queuing Analysis

The compact mode of C-Sec protocol introduces relations between packets; a packet cannot leave the node before the next packet arrives if it is the only packet in the output buffer. This will introduce additional delay overhead that did not exist before for such a packet. To evaluate this delay, simulations were performed to evaluate the end-to-end delay for C-Sec and MiniSec at 20 packets per second, for one thousand seconds. The MiniSec protocol is used for comparison because it has the smallest header size among other protocols and the closest to C-Sec packet.

Figure 4.11 and Figure 4.12 show the histograms for end-to-end packet delay results with 95% confidence. It can be observed that C-Sec introduces additional delay overhead of 200 ms on average, and more packets wait longer time than 800 ms for the C-Sec compared to MiniSec. This is a direct result for the relation between packets introduced by the C-Sec.

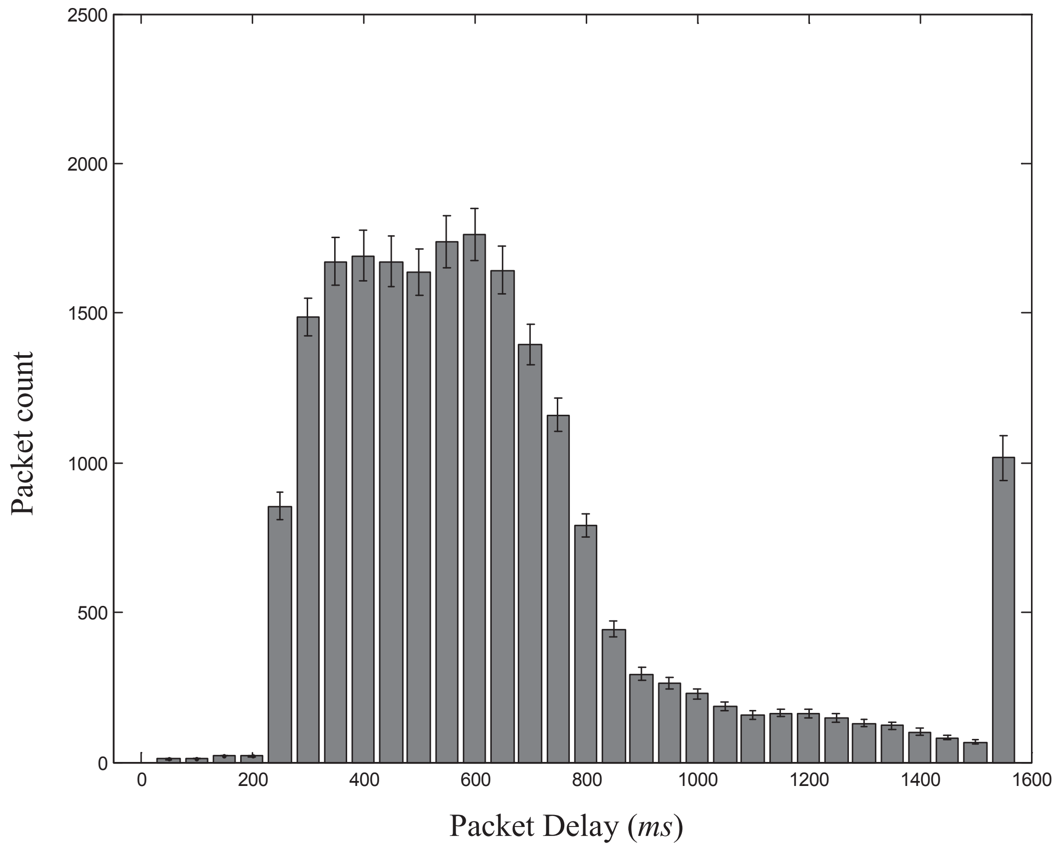


Figure 4.11 Histogram of average end-to-end packet delay per node for C-Sec

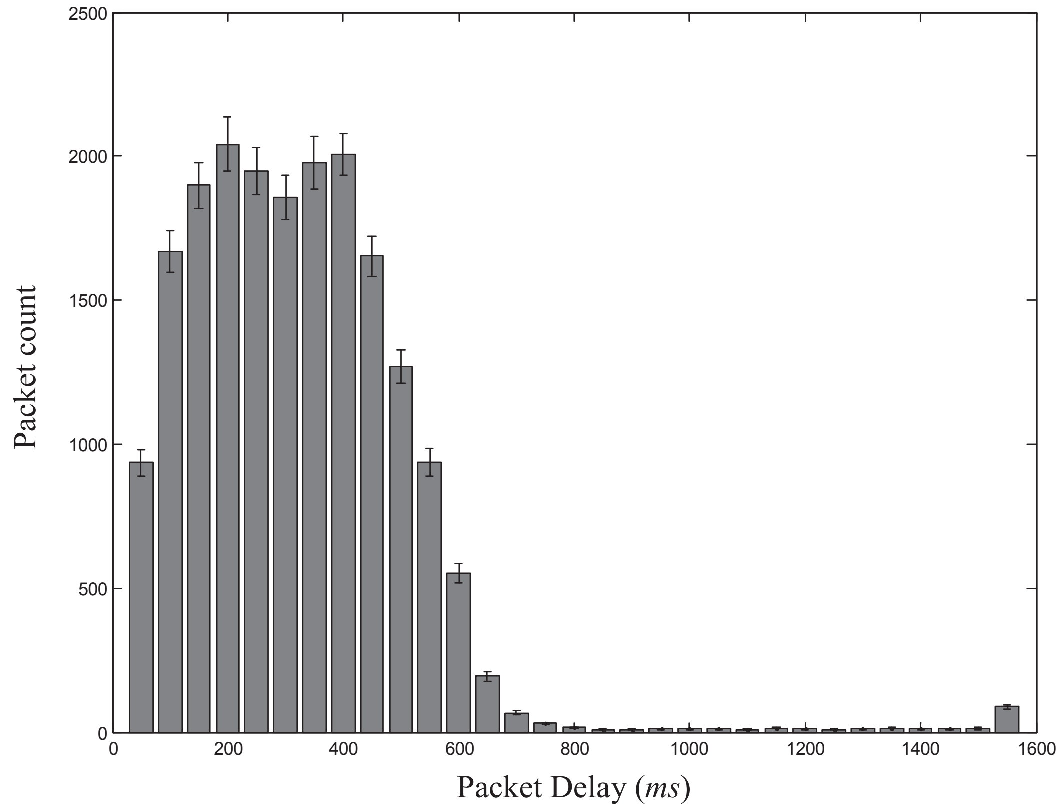


Figure 4.12 Histogram of average packet delay per node MiniSec

To study this delay more deeply, the mathematical queuing model and delay analysis used in traditional protocols like the MiniSec is examined. These protocols follow the M/M/1 queuing model, which represents a system having a single server, where arrivals are determined by a Poisson process and service times have an exponential distribution [108]. This model is compared to a modified model that considers the new condition introduced by the C-Sec; a packet cannot depart until the next packet arrives even if the service is finished. This restriction happens when there is only one packet in the queue.

An M/M/1 queue is described by the following parameters:

λ: Arrival rate, interarrival times are exponentially distributed with a mean of $1/\lambda$.

μ: Departure rate, service times are exponentially distributed with a mean of $1/\mu$.

As Figure 4.13 shows, the C-Sec queuing model can be described as a variation of a continuous time markov chain, which is defined by transition rates between states. The states are the number of packets in the system.

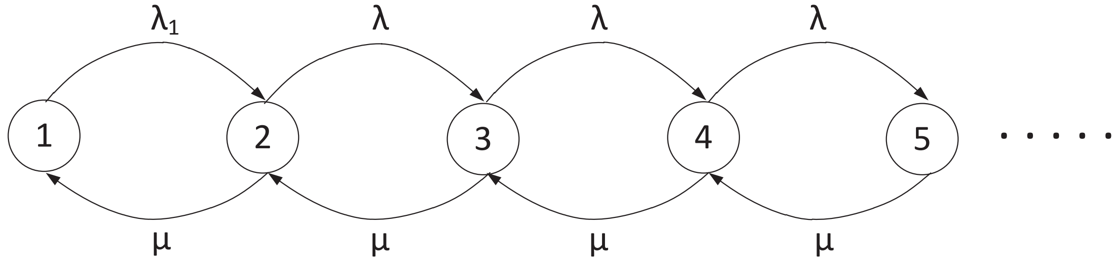


Figure 4.13 Markov Chain for C-Sec

Note that all transition rates from state n to state $n+1$ are λ except for the case where $n = 1$. To compute the transition rate λ_1 for this case, it has to be taken into account whether the packet is being processed or waiting for the next packet to arrive. To solve the Markov chain system in Figure 4.13, let $X(t)$ be the state of the system at time t, and let Δt be a small time interval. Then the probability of transition from state 1 to state 2 is given by:

$$P\big((X(t+\Delta t)=2)\mid(X(t)=1)\big)\cong\lambda_1\Delta t \tag{4.8}$$

The event $\mathrm{X(t)}=1$ is the union of the disjoint events A and B where:

A: The packet is being served.

B: The packet is waiting for the next arrival.

Then Equation 4.8 will be:

$$P\big((X(t+\Delta t)=2)\mid(A\cup B)\big)=\frac{P\big((X(t+\Delta t)=2)\cap(A\cup B)\big)}{P(A)+P(B)}$$

$$=\frac{P\big((X(t+\Delta t)=2)\cap A\big)+P\big((X(t+\Delta t)=2)\cap B\big)}{P(A)+P(B)} \tag{4.9}$$

However, the first packet will not wait in the system if it finished processing and the next packet has arrived, therefore:

$$P\big((X(t+\Delta t)=2)\cap B\big)=0 \tag{4.10}$$

If the first packet is being processed, then it has to wait even if the next packet arrives, then:

$$P\big((X(t+\Delta t)=2)\cap A\big)\ =P\big((X(t+\Delta t)=2|A).P(A)\big)$$
$$=\lambda\Delta tP(A) \tag{4.11}$$

Substituting 4.10 and 4.11 in 4.9:

$$P\big((X(t+\Delta t)=2) \mid (X(t)=1)\big) = \frac{\lambda \Delta t P(A)}{P(A)+P(B)}$$

$$= \frac{\lambda \Delta t P(A)/P(B)}{1+P(A)/P(B)} \tag{4.12}$$

However, there is a relation between $P(A)$ and $P(B)$ because:

$$P(A \text{ at time } t+\Delta t \mid B \text{ at time } t) \cong \lambda \Delta t \tag{4.13}$$

$$P(B \text{ at time } t+\Delta t \mid A \text{ at time } t) \cong \mu \Delta t \tag{4.14}$$

$$\lambda P(B) = \mu P(A)$$

$$\frac{P(A)}{P(B)} = \frac{\lambda}{\mu} = \rho$$

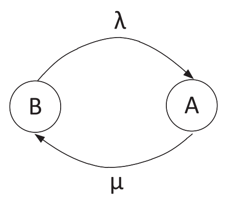


Figure 4.14 Markov chain balance equation

Substituting ρ from the markov-chain balance equation in Figure 4.14 yields:

$$P\big((X(t+\Delta t)=2) \mid (X(t)=1)\big) = \lambda \Delta t \frac{\rho}{1+\rho} \tag{4.15}$$

From 4.8 $\lambda_1$can be computed:

$$\lambda_1 \Delta t = \lambda \Delta t \frac{\rho}{1+\rho}$$

$$\lambda_1 = \lambda \frac{\rho}{1+\rho} \tag{4.16}$$

Let $p_1, p_2, p_3, \ldots$ be the probabilities of each of the states. The global balance equation: $Rate\ in = Rate\ out$ for each state yields:

State 1: $\lambda_1 p_1 = \mu p_2 \quad\quad \rightarrow \quad p_2 = \frac{\rho^2}{1+\rho} p_1$

State 2: $\mu p_3 = \lambda p_2 \quad \rightarrow \quad p_3 = \rho p_2 \quad\quad \rightarrow \quad p_3 = \frac{\rho^3}{1+\rho} p_1$

$\vdots$

State j-1: $\mu p_j = \lambda p_{j-1} \quad \rightarrow \quad p_j = \rho p_{j-1} \quad\quad \rightarrow \quad p_j = \frac{\rho^j}{1+\rho} p_1$

Now:

$$\sum_{j=1}^{\infty} p_j = 1 \tag{4.17}$$

Substituting the probabilities of the states in to 4.16:

$$p_1 + \frac{\rho^2}{1+\rho} p_1 + \frac{\rho^3}{1+\rho} p_1 + \frac{\rho^4}{1+\rho} p_1 + \cdots = 1 \tag{4.18}$$

$$p_1 + \frac{\rho^2 p_1}{1+\rho}(1 + \rho + \rho^2 + \cdots) = 1 \tag{4.19}$$

From 4.19, p_1 can be computed as:

$$p_1 = (1-\rho)(1+\rho) \tag{4.20}$$

Combining 4.17 and 4.20, we get:

$$p_j = \rho^j (1-\rho) \qquad j > 1 \tag{4.21}$$

From 4.20 and 4.21, the expected number of packet in the C-Sec system:

$$E(Number\ of\ packets\ in\ the\ C - Sec\ system) = \sum_{n=1}^{\infty} n p_n$$

$$= (1-\rho)(1+\rho) + \sum_{n=2}^{\infty} n \rho^n (1-\rho)$$

$$= (1 + \rho) + \sum_{n=1}^{\infty} n\rho^n(1 - \rho)$$

$$= (1 - \rho) + \rho(1 - \rho)\sum_{n=1}^{\infty} n\rho^{n-1}$$

$$= (1 - \rho) + \rho(1 - \rho)\frac{d}{d\rho}\sum_{n=0}^{\infty} \rho^n$$

$$= (1 - \rho) + \rho(1 - \rho)\left(\frac{1}{(1 - \rho)^2}\right)$$

$$= 1 - \rho + \frac{\rho}{1 - \rho} \tag{4.22}$$

The expected value for the number of packets in the system for the M/M/1 queuing model is:

$$E(Number\ of\ packets\ in\ the\ M/M/1\ system) = \frac{\rho}{1 - \rho} \tag{4.23}$$

Comparing Equation 4.22 and 4.23, it can be noted that the difference between the queuing models is $1 - \rho$. This value is less than one, because $\rho < 1$. It can be concluded that expected number of packets in the C-Sec system is one packet greater than the M/M/1 system in the worst case and the difference approaches zero when the network is more utilized.

Using Little's law, the expected time in the C-Sec system is:

$$E(time\ in\ the\ C - Sec\ system) = E(packets\ in\ the\ C - Sec\ system)/arrival\ rate$$

$$= \left(1 - \rho + \frac{\rho}{1 - \rho}\right)/\lambda$$

$$= \frac{1}{\lambda} - \frac{1}{\mu} + \frac{1}{\mu - \lambda} \tag{4.24}$$

The expected value of waiting time for the M/M/1 system is given by:

$$E(time\ in\ the\ M/M/1\ system) = \frac{1}{\mu - \lambda} \tag{4.25}$$

Figure 4.15 shows the expected value of waiting time for both the C-Sec and the M/M/1 queuing models. The value of μ mainly depends on the bit rate of the wireless medium i.e. 250-kbps, the size of the data and acknowledgment packets, and the duty cycle of the TMAC protocol. The value of μ is chosen to be 120 packets per second based on maximum of number packets that a node can handle in Castalia simulation. Compared to the expected value of the waiting time for the M/M/1 queue, C-Sec has higher waiting time for low arrival rates, because a single packet in the queue is held until the next packet arrives. However, C-Sec approaches the M/M/1 queue behavior for larger arrival rates because the likelihood of a single packet in the queue is small, which is the only difference between C-Sec and M/M/1 queuing models.

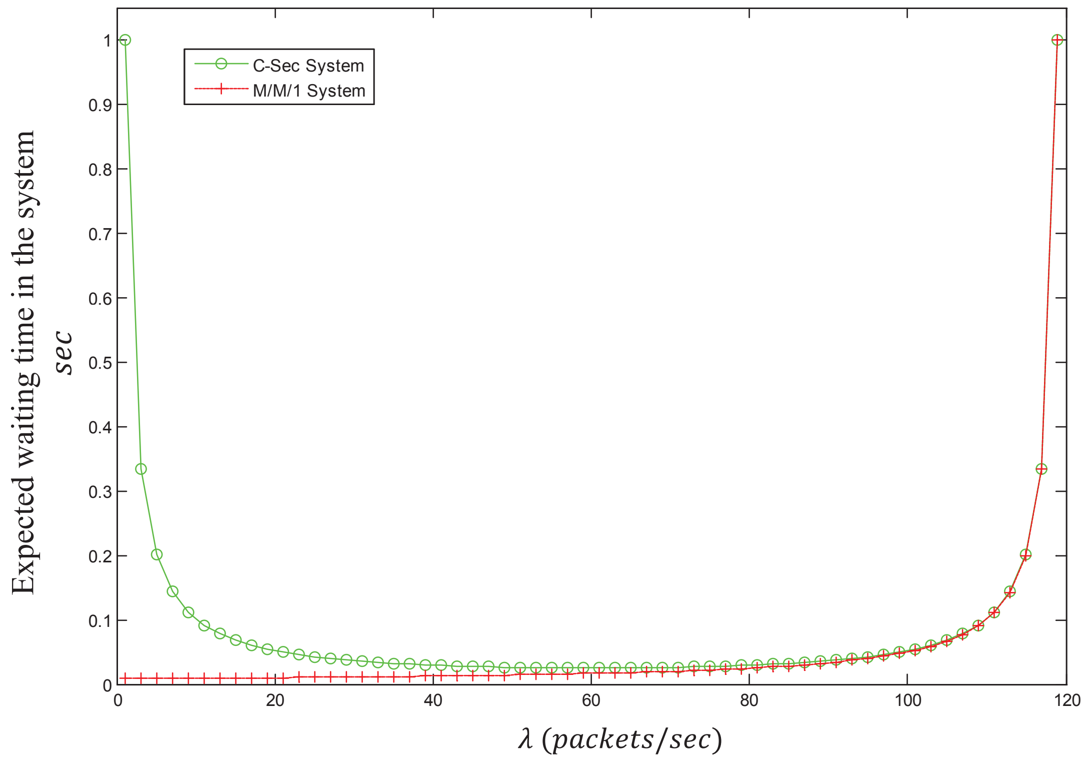


Figure 4.15 Expected value of waiting time in the system (μ=120 packets/sec)

Equation 4.26 shows the difference of expected time in the system between the C-Sec and M/M/1 models, which is the average waiting time introduced by the C-Sec over the classic M/M/1 queuing model.

$$E(time\ in\ the\ C-Sec\ system) - E(time\ in\ the\ M/M/1\ system) = \frac{1}{\lambda} - \frac{1}{\mu} \qquad (4.26)$$

Figure 4.16 compares the value in Equation 4.26 with simulation results of the additional waiting time introduced by C-Sec, i.e. the average period of time the current compact packet waits until the next packet arrives when it is the only packet in the system. It can be inferred that as the traffic load increases, the additional waiting time introduced by the C-Sec queuing model decreases. The two curves have similar trends; however, the additional waiting time introduced by the C-Sec protocol in the simulation results is slightly higher. This is because it is affected by the duty cycle of the T-MAC and packet retransmissions due to bit errors. For lower values of packet arrival time, the simulation results shows less overhead time, that is because packets with waiting times larger than the authentication timer are transmitted in the conventional mode, this puts a cap on the maximum waiting time for compact packets.

Generally, most of WSN applications do not require hard real-time constraints [23], and such additional packet delays can be tolerated. The importance of the energy savings the C-Sec can achieve outweighs the additional packet delay it introduces.

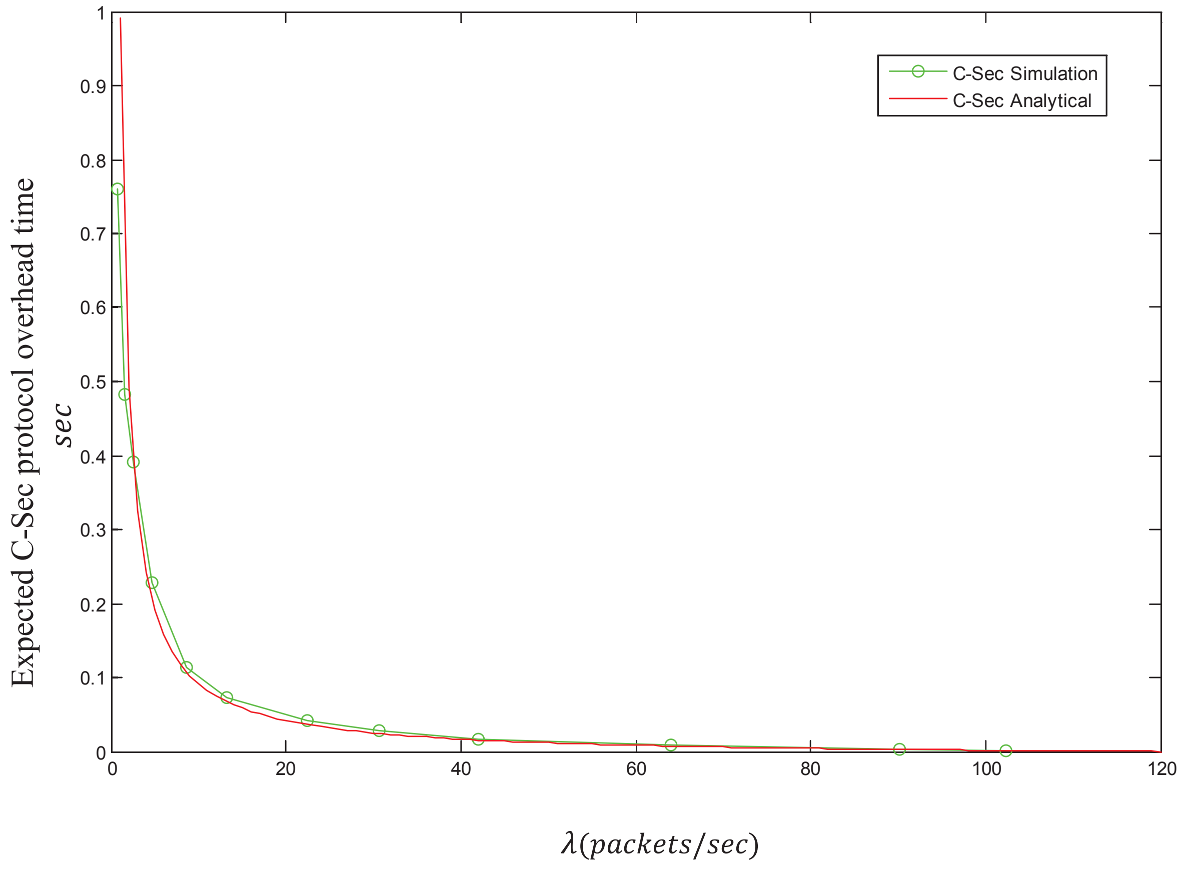


Figure 4.16 Average C-Sec overhead time (μ=120 packets/sec)

4.3.4 Error probability

This section analyzes the impact of the dependency between packets introduced by the C-Sec protocol on error probability. Stochastic models that measure the impact of block cipher encryption on post decryption bit errors [117-119] are extended to measure the impact of packet dependency introduced by C-Sec using Markov characterization stochastic model.

For typical encryption protocols, the packet consists of two parts, the header block and the encrypted message block. If a bit error occurs in the encrypted message block during transmission in the wireless channel, this error will expand to all bits of the decrypted message with a probability of 50%. This is a result of the Strict Avalanche Criterion (SAC) of the encryption algorithms [109]. Some researchers tried to avoid the effect of SAC by inventing new encryption algorithms like [110] [122]. Such algorithms trade off security with performance and are not scrutinized enough to be used for wireless communication.

The packet can be divided in to two parts, an encrypted body that is affected by the SAC and the header. If a bit error occurs in the encrypted body of the message, it will expand as a result of the SAC effect. However, if it happens in the header of the message, only that bit will be changed in the receiver side. As shown in Figure 4.15, this will result in four different possibilities for the bit error.

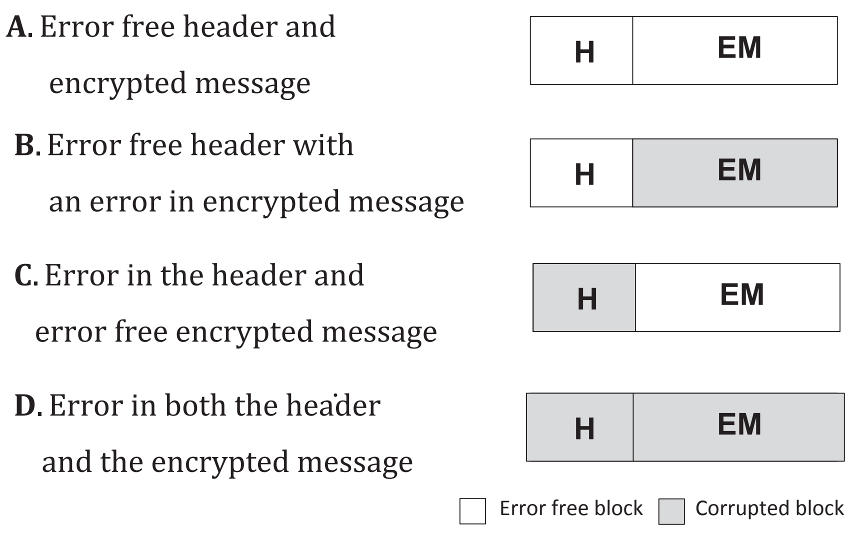

A. Error free header and encrypted message
H
EM
B. Error free header with an error in encrypted message
H
EM
C. Error in the header and error free encrypted message
H
EM
D. Error in both the header and the encrypted message
H
EM
Error free block
Corrupted block

Figure 4.17 Error event states

Let b_i denote the i-th bit of the header block. $\forall\, i = 1 \ldots H$. Let $b_1 b_2 b_3 \ldots b_H$ be the transmitted header bits, and $b'_1 b'_2 b'_3 \ldots b'_H$ be received header bits.

Let $P(b'_i|b_i)$ denote the probability of receiving b'_i when b_i is transmitted, and $P\left(b'_1 b'_2 \ldots b'_{H_l} | b_1 b_2 \ldots b_H\right)$ denote the probability that that the received header is

$b_1'b_2' \dots b_H'$ when the transmitted header is: $b_1 b_2 \dots b_H$. Assuming reception of each bit is independent of all the remaining bits, then:

$$P\left(b_1'b_2' \dots b_{N_c}' | b_1 b_2 \dots b_{H_c}\right) = P(b_1'|b_1).P(b_2'|b_2)\dots P(b_H'|b_H) \tag{4.27}$$

If, in a certain received block, i bits are in error, then $H - i$ bits are correct. Then the probability of receiving this block will be $P_{bit}^{i}(1 - P_{bit})^{H-i}$. There are $\binom{H}{i}$ different ways in which i errors can occur in H bits. Hence, if the header of the message is sent and it contains *i* bits that had errors, then $H - i$ bits are correct. Then, the probability of receiving a header with i errors is:

$$\begin{aligned} P(i - errors\ in\ the\ header) &= \binom{H}{i} P_{bit}^{i}(1 - P_{bit})^{H-i} \\ &= \binom{H}{i} P_{bit}^{i} {Q_{bit}}^{H-i} \end{aligned} \tag{4.28}$$

Where $Q_{bit} = (1 - P_{bit})$. The probability of a receiving correct header:

$$P(header\ is\ correct) = \binom{H}{0} P_{bit}^{0} {Q_{bit}}^{H-0} = {Q_{bit}}^{H} \tag{4.29}$$

Similarly, the probability of receiving a correct encrypted message payload:

$$P(EM\ is\ correct) = {Q_{bit}}^{EM} \tag{4.30}$$

Depending on the state of the received header and message body at each decryption cycle, the decryptor node can be in one of the four states described in Figure 4.15. The probability of receiving a message with a correct header and body is:

$$\begin{aligned} P(A) &= P(header\ is\ correct) * P(EM\ is\ correct) \\ &= {Q_{bit}}^{H} * {Q_{bit}}^{EM} \end{aligned} \tag{4.31}$$

Similarly:

$$P(B) = \left(1 - Q_{bit}{}^{EM}\right) * Q_{bit}{}^{H} \tag{4.32}$$

$$P(C) = Q_{bit}{}^{EM} * \left(1 - Q_{bit}{}^{H}\right) \tag{4.33}$$

$$P(D) = \left(1 - Q_{bit}{}^{EM}\right) * \left(1 - Q_{bit}{}^{H}\right) \tag{4.34}$$

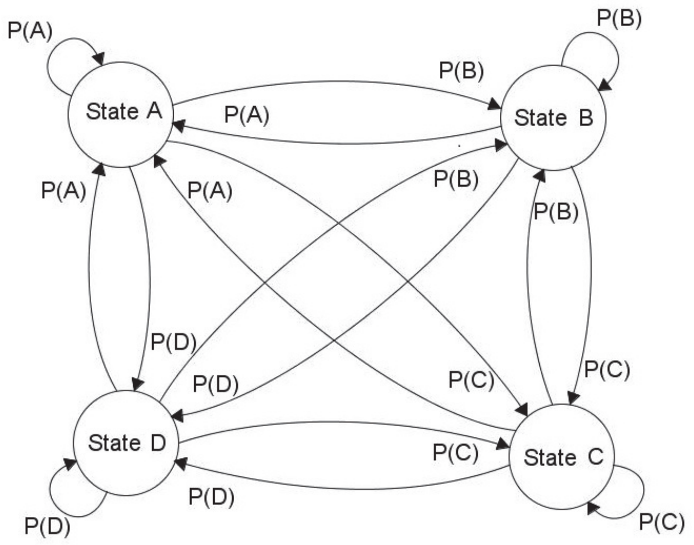


Figure 4.18 The state diagram for error events at decryption process

Using these states, the state transition diagram in Figure 4.16 is created. The associated transition probability matrix to denote the probability of moving from state i to j is given as follows:

$$P_{i,j} = \begin{pmatrix} P(A) & P(B) & P(C) & P(D) \\ P(A) & P(B) & P(C) & P(D) \\ P(A) & P(B) & P(C) & P(D) \\ P(A) & P(B) & P(C) & P(D) \end{pmatrix} \tag{4.35}$$

Then, the mean probability of error P_{error} can be computed as:

$$P_{error} = P(A)e_1 + P(B)e_2 + P(C)e_3 + P(D)e_4 \tag{4.36}$$

Where $e_1, e_2, e_3,$ and e_4 are the error rates associated with each of the states in Figure 4.16. The value of e_1 is zero, because the packet is error free at state A. The value of e_2 is 0.5 because in state B the decrypted body of the packet follows the SAC. The value of e_3 is equal to P_{bit} because errors occur in the header at state C. Assuming that the bit errors events in the header and bit errors in the decrypted message are independent, the bit error rate in state D can be computed as:

$$\begin{aligned} e_4 &= 1 - (1 - 0.5)(1 - P_{bit}) \\ &= 0.5(1 + P_{bit}) \end{aligned} \tag{4.37}$$

Then the post decryption probability of error will be:

$$\begin{aligned} P_{error} = P_{bit}(1 - Q_{bit}{}^{H})Q_{bit}{}^{EM} + 0.5Q_{bit}{}^{H}(1 - Q_{bit}{}^{EM}) + \\ 0.5(1 + P_{bit})(1 - Q_{bit}{}^{H})(1 - Q_{bit}{}^{EM}) \end{aligned} \tag{4.38}$$

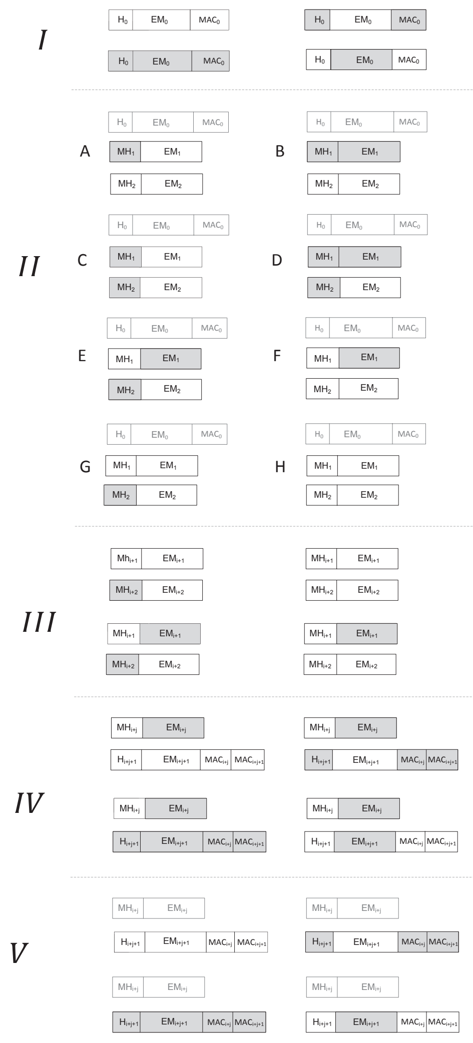
For C-Sec, the relations between packets introduce dependency in the error probability. To model this dependency, all possible packets formats with different error events associated with them are considered. This includes five different pairings of dependency between packets with different formats. The error events associated with these cases, labeled A to E, are shown in Figure 4.17. The shaded fields in the packets represent fields with errors. The post decryption probability of error P_{error} is computed for each case in a similar manner, taking in to account the dependency in a stream of packets as follows:

I. The first packet is independent conventional, it follows the traditional post decryption error probability model.

II. The second packet is independent from the first packet but it has dependency on the masked header of the third packet. As shown in Figure 4.17 –B, this include eight error events.

A. Error in the header of the second packet, error free encrypted message of the second packet, and error free header of the third packet. The probability of this event is:

$$P(A) = \left(1 - Q_{bit}{}^{H}\right) * Q_{bit}{}^{EM} * Q_{bit}{}^{H} \tag{4.39}$$



$$P_{error} = P_{bit}\left(1 - Q_{bit}{}^{H+MAC}\right)Q_{bit}{}^{EM} + 0.5Q_{bit}{}^{H+MAC}\left(1 - Q_{bit}{}^{EM}\right) + 0.5(1 + P_{bit})\left(1 - Q_{bit}{}^{H+MAC}\right)\left(1 - Q_{bit}{}^{EM}\right) \tag{4.40}$$

$$P_{error} = 2P_{bit}\left(1 - Q_{bit}{}^{MH}\right)Q_{bit}{}^{EM+MH} + Q_{bit}{}^{EM}\left(P_{bit}\left(1 - Q_{bit}{}^{MH}\right)\right)^{2} + 0.5\left((1 + P_{bit})\left(1 - Q_{bit}{}^{MH}\right)\right)^{2}\left(1 - Q_{bit}{}^{EM}\right) 0.5Q_{bit}{}^{2MH}\left(1 - Q_{bit}{}^{EM}\right) + Q_{bit}{}^{MH}(1 + P_{bit})\left(1 - Q_{bit}{}^{MH}\right)\left(1 - Q_{bit}{}^{EM}\right) \tag{4.41}$$

$$P_{error} = P_{bit}\left(1 - Q_{bit}{}^{MH}\right)Q_{bit}{}^{MH} + 0.5Q_{bit}{}^{MH}\left(1 - Q_{bit}{}^{EM}\right) + 0.5(1 + P_{bit})\left(1 - Q_{bit}{}^{MH}\right)\left(1 - Q_{bit}{}^{EM}\right) \tag{4.42}$$

$$P_{error} = 0.5(1 + P_{bit})\left(1 - Q_{bit}{}^{2MAC+H}\right)\left(1 - Q_{bit}{}^{EM}\right) + 0.5Q_{bit}{}^{2MAC+H}\left(1 - Q_{bit}{}^{EM}\right) + P_{bit}\left(1 - Q_{bit}{}^{2MAC+H}\right)\left(1 - Q_{bit}{}^{EM}\right) \tag{4.43}$$

$$P_{error} = P_{bit}\left(1 - Q_{bit}{}^{2MAC+H}\right)Q_{bit}{}^{EM} + 0.5\left(1 - Q_{bit}{}^{EM}\right)Q_{bit}{}^{H+2MAC} + 0.5(1 + P_{bit})\left(1 - Q_{bit}{}^{H+2MAC}\right)\left(1 - Q_{bit}{}^{EM}\right) \tag{4.44}$$

Figure 4.19 Various error event states and related mean error probability equations for C-Sec protocol

B. Error in the header and encrypted message of the second packet, and error free header of the third packet. The probability of this event is:

$$P(B) = \left(1 - {Q_{bit}}^{H}\right) * \left(1 - {Q_{bit}}^{EM}\right) * {Q_{bit}}^{H} \tag{4.45}$$

C. Error in the header of the second packet, error free encrypted message of the second packet, and error in the header of the third packet. The probability of this event is:

$$P(C) = \left(1 - {Q_{bit}}^{H}\right)^2 * {Q_{bit}}^{EM} \tag{4.46}$$

D. Error in the header and encrypted message of the second packet, and error in the header of the third packet. The probability of this event is:

$$P(D) = \left(1 - {Q_{bit}}^{H}\right)^2 * \left(1 - {Q_{bit}}^{EM}\right) \tag{4.47}$$

E. Error free header of the second packet, error in the encrypted message of the second packet and error in the header of the third packet. The probability of this event is:

$$P(E) = \left(1 - {Q_{bit}}^{H}\right) * \left(1 - {Q_{bit}}^{EM}\right) * {Q_{bit}}^{H} \tag{4.48}$$

F. Error free header of the second packet and third packet, error in the encrypted message of the second packet. The probability of this event is:

$$P(F) = \left(1 - {Q_{bit}}^{EM}\right) * {Q_{bit}}^{2H} \tag{4.49}$$

G. Error free header and encrypted message of the second packet, error in the header of the third packet. The probability of this event is:

$$P(G) = \left(1 - {Q_{bit}}^{H}\right) * {Q_{bit}}^{EM} * {Q_{bit}}^{H} \tag{4.50}$$

H. Error free header and encrypted message of the second packet, error free header of the third packet. The probability of this event is:

$$P(H) = {Q_{bit}}^{EM} * {Q_{bit}}^{2H} \tag{4.51}$$

Then, the mean probability of error P_{error} for these cases can be computed as:

$$P_{error} = P(A)e_1 + P(B)e_2 + P(C)e_3 + P(D)e_4 + P(E)e_5 + P(F)e_6 + P(G)e_7 + P(H)e_8 \tag{4.52}$$

The value of error rates e_1 , e_3, and e_7 is P_{bit} because errors do not propagate in the associated events as they occur in the headers of the packets. The e_2, e_4 and e_5 involve errors in the header and encrypted message, assuming that channel bit errors and errors caused by SAC occur independently, they can be computed as:

$$= 1 - (1 - 0.5)(1 - P_{bit})$$

$$= 0.5(1 + P_{bit}) \tag{4.53}$$

The value of e_6 is 0.5 because the error occurs in the encrypted message which follows the SAC, and e_7 is zero hence no errors occur. Substituting the error rates in Equation 4.47 will yield Equation 4.50 in Figure 4.17.

III. The header of the packet $\mathrm{i} + 1$ is already verified, it has dependency on the header of the packet $i + 2$. As for II the value of P_{error} is computed for this case. The result is shown in Equation 4.51.

IV. The last compact packet depends on the header of the last conventional packet, which has different size. The value of P_{error} for this case is shown in Equation 4.52.

V. The last conventional packet is similar to the first conventional packet, but with different header size. The value of P_{error} for this case is shown in Equation 4.53.

To evaluate the performance of post decryption error probability for the C-Sec, a weighted average of cases given by Equations 4.51- 4.55 is computed assuming a 96% of compact packets. This value matches the percent of compact packets with a payload size of 30 bytes given in the simulation results in Section 4.3.1. The post-decryption error probability of the MiniSec protocol is given by Equation 4.36. The MiniSec is used to compare with because it has the smallest packet size among other protocols and the closest to C-Sec packet. Figure 4.18 shows the plots of post-decryption error probability for both cases. The difference in output error probability is negligible and it does not exceed 1%.

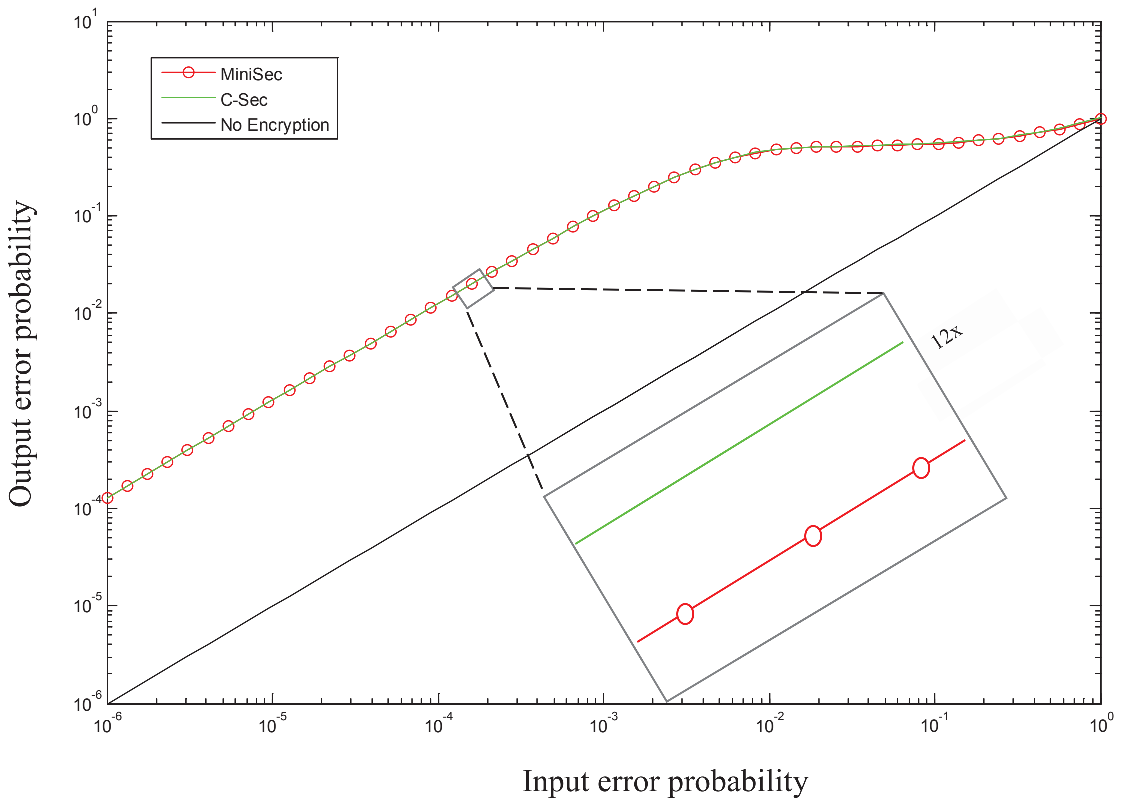


Figure 4.20 Post decryption error probabilities for C-Sec and MiniSec

Figure 4.19 shows the percentages of packet loss, with 95% confidence, for C-Sec and MiniSec protocols, obtained from simulations using Castalia with varying noise floor (the sum of all the noise sources and unwanted signals). The packet loss increases with the increase of noise floor. The C-Sec has about 1% more dropped packets than the Minisec in the active region. Compared to the high energy gain of the C-Sec, this amount of packet loss is can be probably be tolerated in many applications. As mentioned in the protocol description, the used of conventional mode can be enforced in case of high QoS constraints or the use of packet loss sensitive applications.

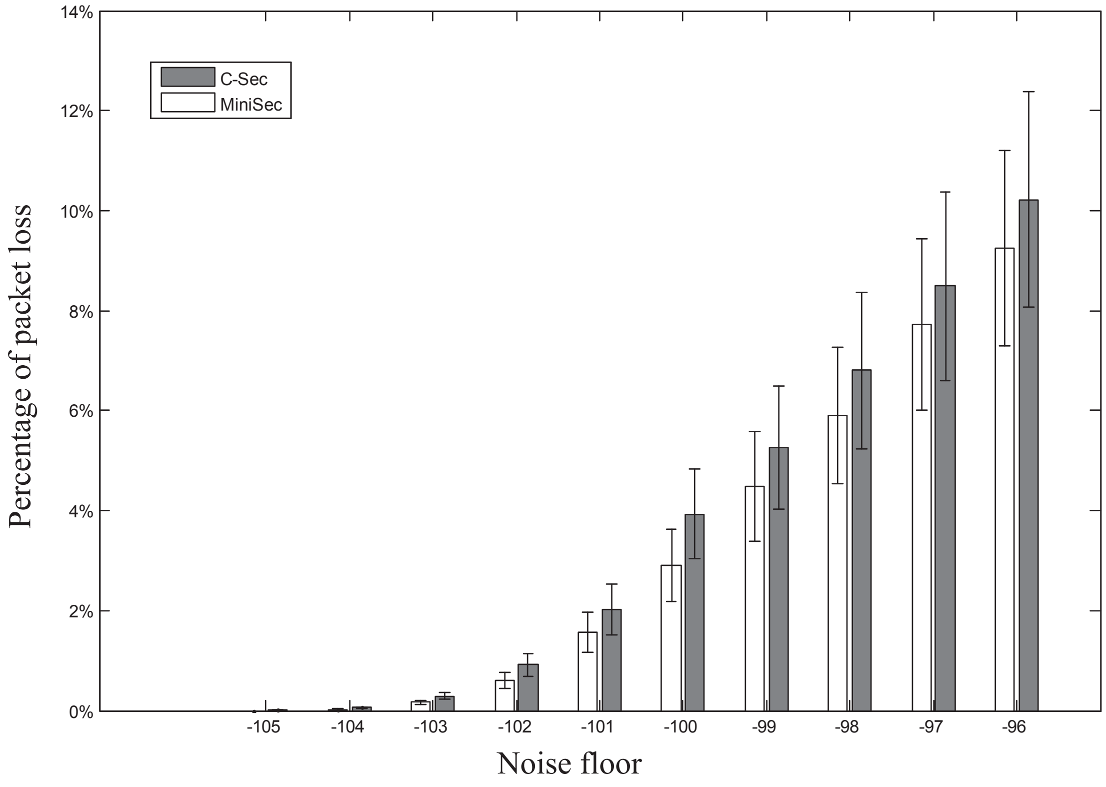


Figure 4.21 Packet loss

4.4 CONCLUDING REMARKS

This chapter provided a detailed description of the C-Sec protocol including the packet formats for the conventional and compact modes, and the procedure of running them on the sender and receiver sides. The evaluation of the security services C-Sec can provide shows that the C-Sec does include all the security services provided by other protocols. In addition, it hides the packet header in the compact mode, which makes it more difficult to eavesdrop on the flow of wireless communication. It also reduces the effect of message spoofing and replay attacks and works as an attack detection mechanism.

To evaluate C-Sec simulations using Castalia simulator were performed for C-Sec and other related protocols with various packet sizes and loads. Simulation results show that C-Sec saves more energy than all related protocols and the TinyOS under different packet loads and packet sizes. The saving in energy can reach up to 19.1% compared to MiniSec, the encryption protocol with the closest packet size to C-Sec.

Both the mathematical analysis of the C-Sec queuing model and the simulation results of the additional overhead time introduced by C-Sec show that, as the traffic load increases, the additional waiting time introduced by the C-Sec queuing model decreases. They also show that C-Sec time overhead increases with lower traffic leads. However, this increase is capped, because packets with waiting times larger than the authentication timer are transmitted in the conventional mode. The additional packet delay overhead of C-Sec can be tolerated, as most of WSNs applications do not require hard real-time constraints. The significance of the energy savings the C-Sec can achieve outweighs the additional packet delay it introduces.

Mathematical analysis of the post-decryption error probability of the C-Sec shows that it slightly increases the post-decryption error probability. Compared to MiniSec, the protocol smallest packet size among related protocols and the closest to C-Sec, the difference in post-decryption error probability does not exceed 1% in the active region. Simulation results show that packet loss increases with the increase of noise floor. However, C-Sec has about 1% more dropped packets than the Minisec in the active region. Compared to the high-energy gain of C-Sec, this amount of packet loss is can be tolerated.

CHAPTER 5 SN-Sec: SENSOR NODE SECURE PLATFORM

Security was not considered when current wireless sensor nodes were designed. As a result, providing a high level of security on current WSNs platforms is unattainable, especially against attacks based on key theft and node compromise. This chapter scrutinizes the security vulnerability in current WSN platforms and compares the main approaches to implementing their cryptographic primitives in terms of security, time, and energy efficiency. To address security weaknesses in other platforms and provide more efficiency SN-Sec is proposed, a 32-bit RISC secure wireless sensor platform with hardware cryptographic primitives.

The SN-Sec provides a system on chip solution that is invulnerable against board-level attacks like bus probing. It is also immune to non-invasive and semi-invasive attacks, and provides unique key storage that is independent from the OS; this means that even if the node OS is compromised, the keys inside the key storage will not be resolved. Providing efficient hardware implementation of asymmetric encryption facilitates energy efficient solutions for key management, certification, and broadcast authentication. Solutions for these problems in the literature tend to avoid asymmetric encryption because of its computational cost. However, these solutions usually have high communication and storage cost that exceed the computational cost of our design.

The choice of cryptographic primitives for SN-Sec is based on their compatibility with the constrained nature of WSNs and their security. The proposed methodology of implementing the encryption primitives for SN-Sec provides protection against power analysis and timing attacks. SN-Sec platform is designed using VHDL. Experimental results using synthesis for Spartan-6 low-power FPGA shows that the proposed design has a very reasonable computational time and energy consumption compared to well-known WSN processers. Being implemented on FPGA makes it more difficult to reverse engineer it and make it more secure compared to microcontroller-based devices [22] used with other WSN platforms.

This chapter is organized as follows: approaches to implementing WSN security platforms, their advantages, disadvantages, and security holes are reviewed in Section 5.1. SN-Sec secure hardware platform is introduced in Section 5.2. Section 5.3 presents the designs of the security primitives and other components used for SN-Sec platform with experimental energy, time and hardware area results using synthesis for Spartan-6 low-power FPGA for each component and for the final SN-Sec platform design. Finally, the concluding remarks are presented in section 5.4.

5.1 Security Weaknesses of Current WSN Platforms

Many researchers proposed mechanisms to provide security for WSNs in the literature. These mechanisms are usually based on underlying implementations of encryption primitives. Regardless of the effectiveness of these mechanisms, they will not provide sufficient security if implemented on unsecure platforms. The current WSN platforms are being made using "off the shelf" components designed for other environments. Secure design for unique WSN environments was not considered in most of the cases.

Many researchers in the literature have addressed efficient implementations of security primitives for WSNs [20] [68-71] [98-104] [113]. Generally, such primitives include public key primitives, private key primitives, and hash functions. These implementations are either software-oriented, or hardware-oriented. The main advantage of software-oriented implementations is their flexibility, because changing them does not require any modification in the hardware architecture of wireless sensor nodes. However, these implementations have large processing overheads on WSN controllers with low processing capacity.

Table 5.1 shows the energy consumption and execution time of elliptic curve based operations implemented in software on the Mica2 wireless node [123]. The relatively high processing time could result in missing some events or not reporting them in a timely manner. It also affects the execution of other programs running on the node, especially if a non-preemptive operating system, like TinyOS, is used. In addition, that consumes large amount of energy compared to hardware implementations up to a factor of 10^3 [123].

Table 5.1 Energy and time results for elliptic curve based operations using software implementations on Mica2 wireless node [123].

Operation	*Key Generation*	*ECDSA signature*	*ECDSA verification*	*D-H key exchange*	*El-Gamal encryption*	*El-Gamal decryption*
Time (*s*)	6.74	6.88	24.17	17.28	24.07	17.87
Energy (*mJ*)	101.1	103.2	362.6	259.2	361.1	268.1

Software implementations are not only less efficient in terms of energy and time, but also are less secure. As Figure 5.1 illustrates, the security of software implementations is dependent on the security of the OS and the underlying wireless node hardware platform, neither of which was designed to be secure in most current WSN platforms.

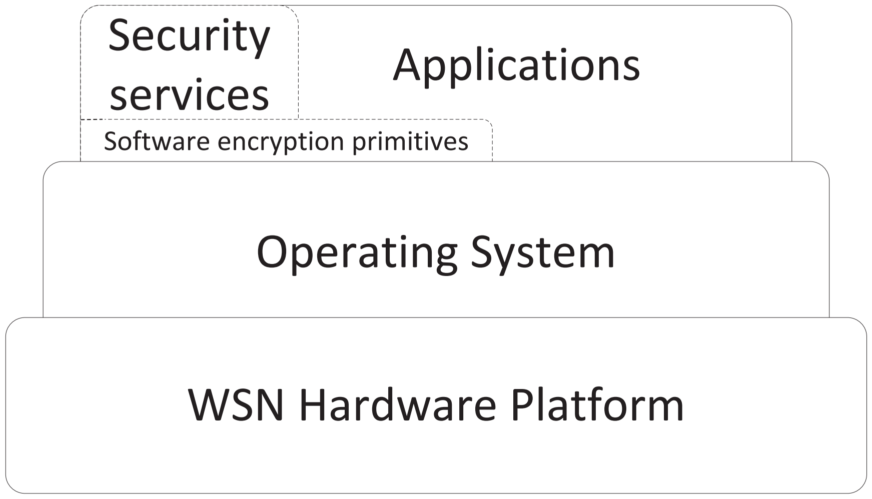


Figure 5.1 Wireless node design using software cryptography over unsecure platform

Because software implementations are run on top of the OS and share memory space with other running programs, they are vulnerable to ease of modification and compromising keys. A clear example of such attacks is the Cache Collision Timing Attacks [124] on software implementations of the AES algorithm. This attack is based on exploiting the characteristics of AES table lookups. It requires direct observation of memory before and after encryption by running the attacking process and the AES algorithm on the same OS.

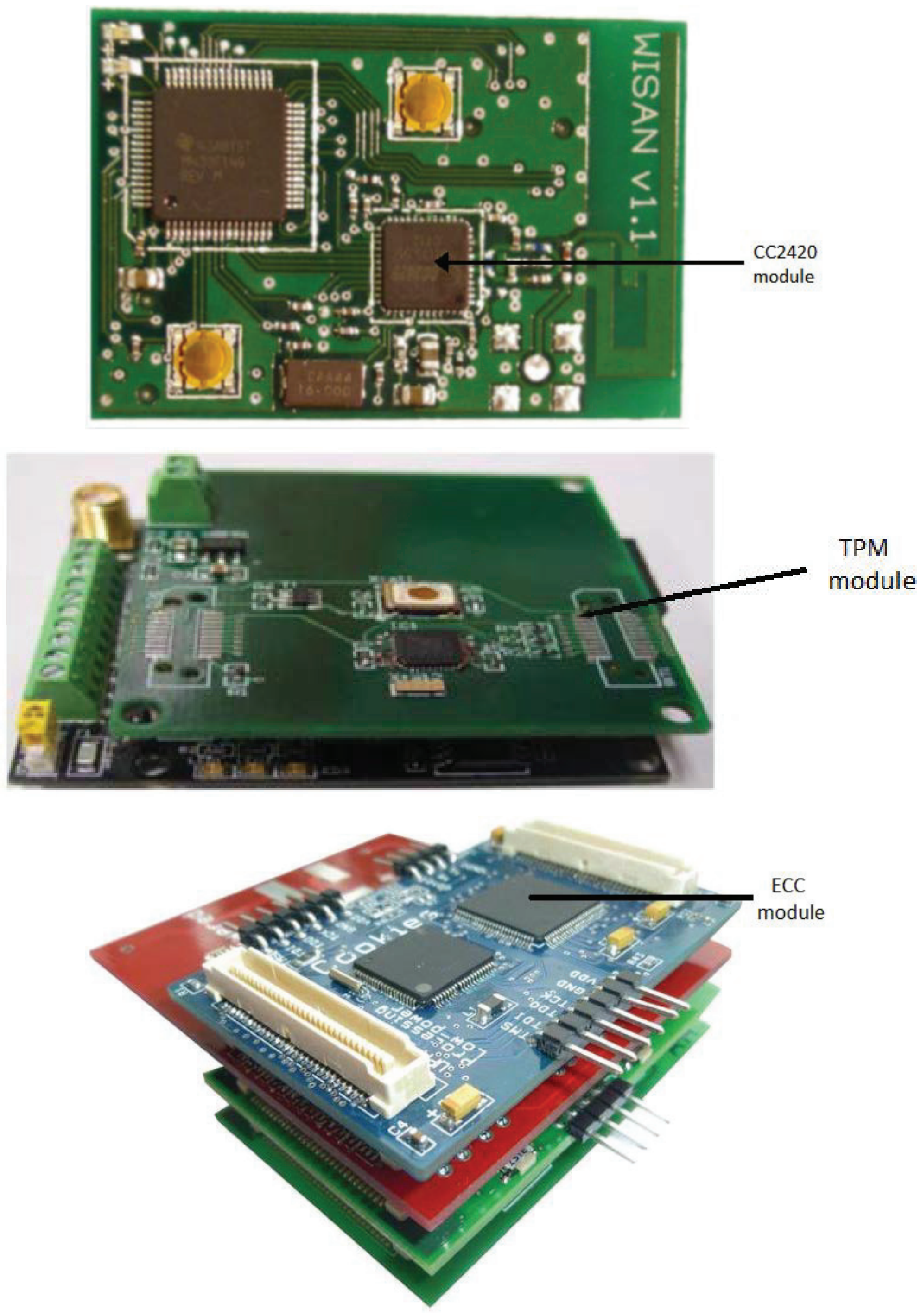


Figure 5.2 Examples of non-secure use of encryption hardware. A) The CC2420 module on the WISAN sensor node. B) The TPM module used with SecFleck C) The ECC module used with the Cookie node

Compared to software implementations of cryptographic algorithms, hardware implementations are not only much faster, but they also consume less energy up to a factor of 10^{-3} [123]. Many sensor platforms use "off the shelf" hardware components to provide security services for WSNs. Figure 5.2 provides such examples, like the use of the Chipcon CC2420 transceiver chip [125] that includes a hardware implementation of the AES algorithm in WISAN sensor mote [19], the use of commodity Trusted Platform Module (TPM) chip for RSA public key encryption in the SecFleck platform [19], and the use of the FPGA based elliptic curve processor in the Cookie node [20].

In all the platforms that use independent hardware to implement security primitives, the communication between the security hardware and other wireless node hardware components is done in plaintext. Launching attacks like bus probing can easily reveal valuable data such as the encryption keys. Other attacks on hardware like Cold Boot Attack [126] can also be used to get the content of the memory and resolve the encryption keys, since the keys are stored in plaintext. Software attacks through the OS are possible as well, because the keys are accessible by the OS in plaintext. Figure 5.3 illustrates the security view of these implementations.

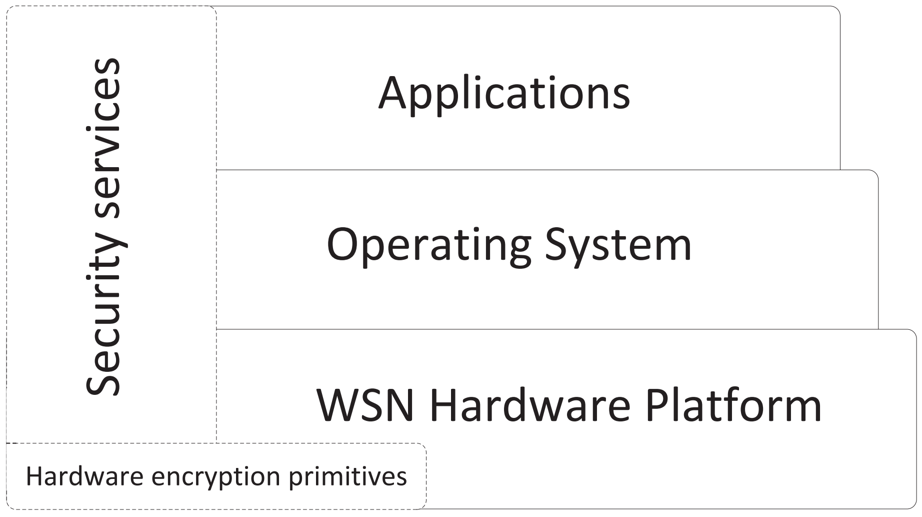


Figure 5.3 Wireless node design using hardware cryptography over unsecure platform and OS

All of the weaknesses and vulnerabilities of previously used schemes will be covered with the SN-Sec platform design. The security view of proposed scheme is shown in Figure 5.4. This scheme aims to redesign the wireless sensor platform from scratch with the security as the main design goal. Many practices can easily help avoid the security faults of other platforms. Designing a system on chip solution helps avoid most of the board-level attacks. In addition, it reduces the cost of having different components on the sensor node board and reduces the energy and latency costs compared to multi-components systems.

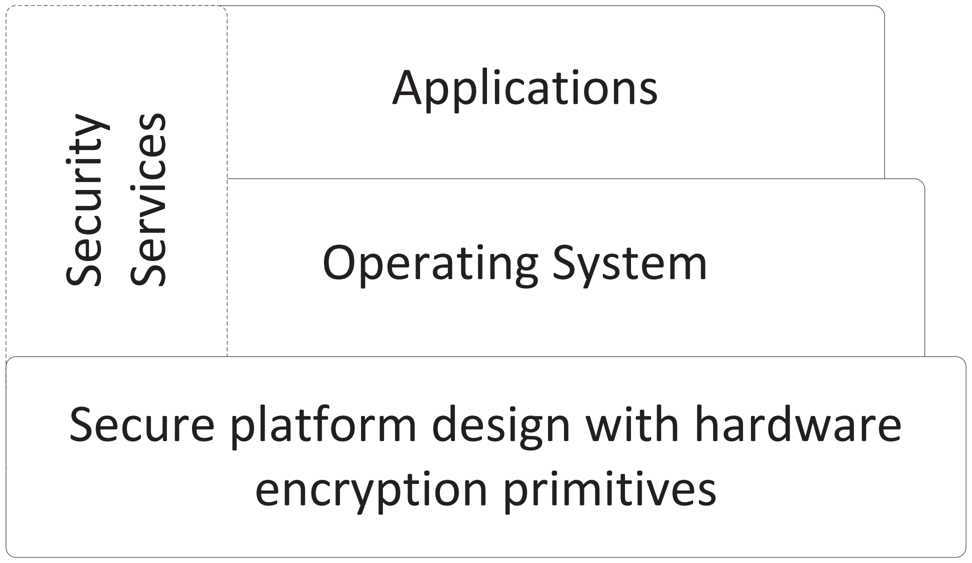


Figure 5.4 Secure node platform design with hardware encryption primitives

Implementing encryption primitives in hardware from scratch facilitate choosing and implementing the best configurations and designs of encryption primitives that meets the constraint nature of WSNs; it frees the main processor from computationally expensive encryption primitives. It also enables implementing secure key storage and encrypted off-chip memory. Including asymmetric primitives helps solving many problems related to key management and distribution and digital signatures

The cost of designing and additional hardware area needed for the security platform will increase manufacturing cost. However, combining all components in one chip, and producing a robust design that is reliable enough to be used in a mass production manner is capable of reducing this cost.

5.2 SN-Sec: Secure Hardware Platform Design

To avoid design weaknesses in previous approaches and guarantee maximum security and energy efficiency for WSNs, SN-Sec is proposed; a secure hardware platform design that provides hardware-based encryption and secure key storage. This design not only avoids high processing overhead and energy consumption of the software implementations, but also provides an advantage over other related hardware implementations because of the careful selection of encryption primitives and their efficient implementations. It also avoids the security weaknesses in previous approaches by being closed and independent from the operating system security.

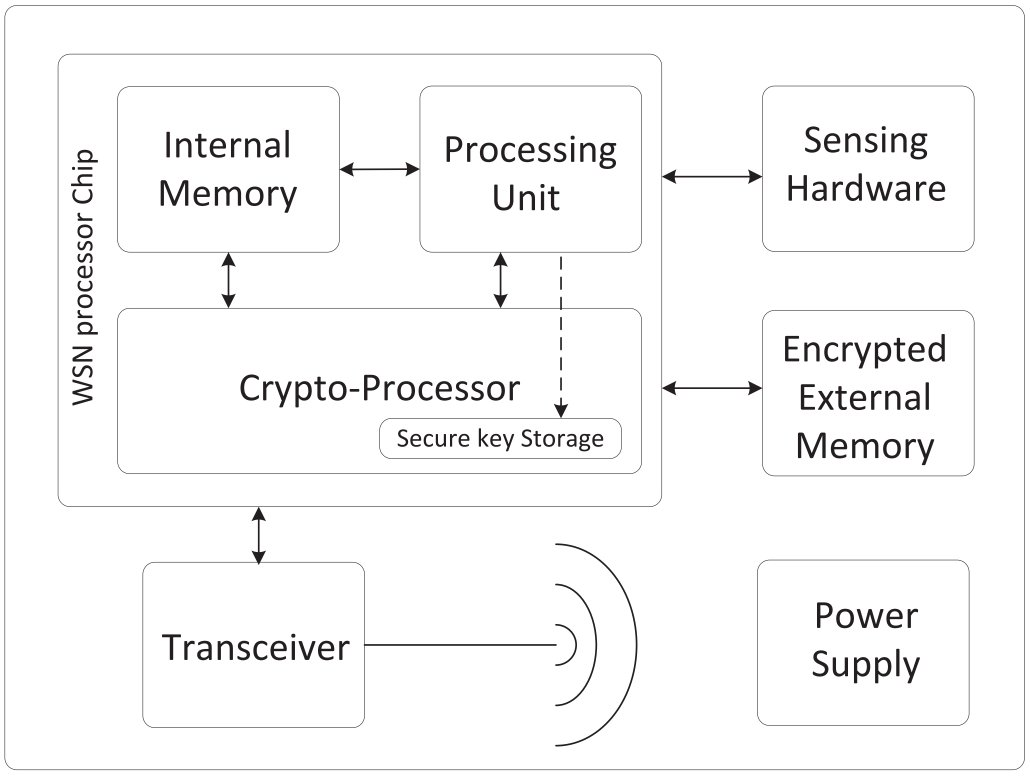


Figure 5.5 Proposed secure platform design

The suggested hardware secure platform for WSNs is shown in Figure 5.5. The sensor node processor unit consists of three main components: the main processor, internal memory, and a crypto-processor. The only part of the data that can be in plaintext is the data within the scope of the sensor node processor chip, like the data being processed inside the main processor and the internal memory. All the other data outside the scope of the sensor processor chip is encrypted, including external memory, communication between WSN components on board like the transceiver, and the traffic in the wireless medium.

5.3 SN-Sec Implementation

The selection of cryptographic primitives for SN-Sec hardware platform should comply with WSN constraints. The implementation of these primitives should be optimized for energy per encryption, power consumption, and hardware area. Speed and throughput were the optimization goals for many implementations, these optimization goals are not considered for WSNs because of their sparse and loose real-time communication. This section evaluates the most efficient configurations and implementation methods of the selected primitives, their security status, and reviews their specialized implementations for WSNs in recent literature.

5.3.1 Advanced Encryption Standard

As mentioned in Section 3.3.1, AES is the best suited algorithm for hardware implementations for WSNs, from security, energy efficiency, latency and hardware area perspectives. It is also shown that the most efficient design for WSN constraints is the 8-bit data-path design with two S-boxes [70], and the most efficient design for the S-box is in the $GF(((2^2)^2)^2)$ composite field [66]. The AES implemented for the SN-Sec platform uses a combination of these designs.

The data-path of this design is shown in Figure 5.6. The design uses two S-boxes one for the key expansion unit that generates a round key on the fly for each round of the algorithm and another for the byte substitution transformation of the round. The design includes a mix-column multiplier that computes the mix-column transformation and produce 128-bit output at each round. This output is converted again to the 8-bit data-path size using a parallel to serial converter unit. One round of the AES algorithm takes 16 cycles using this design. The AES algorithm runs for 10 rounds which requires 160 cycles.

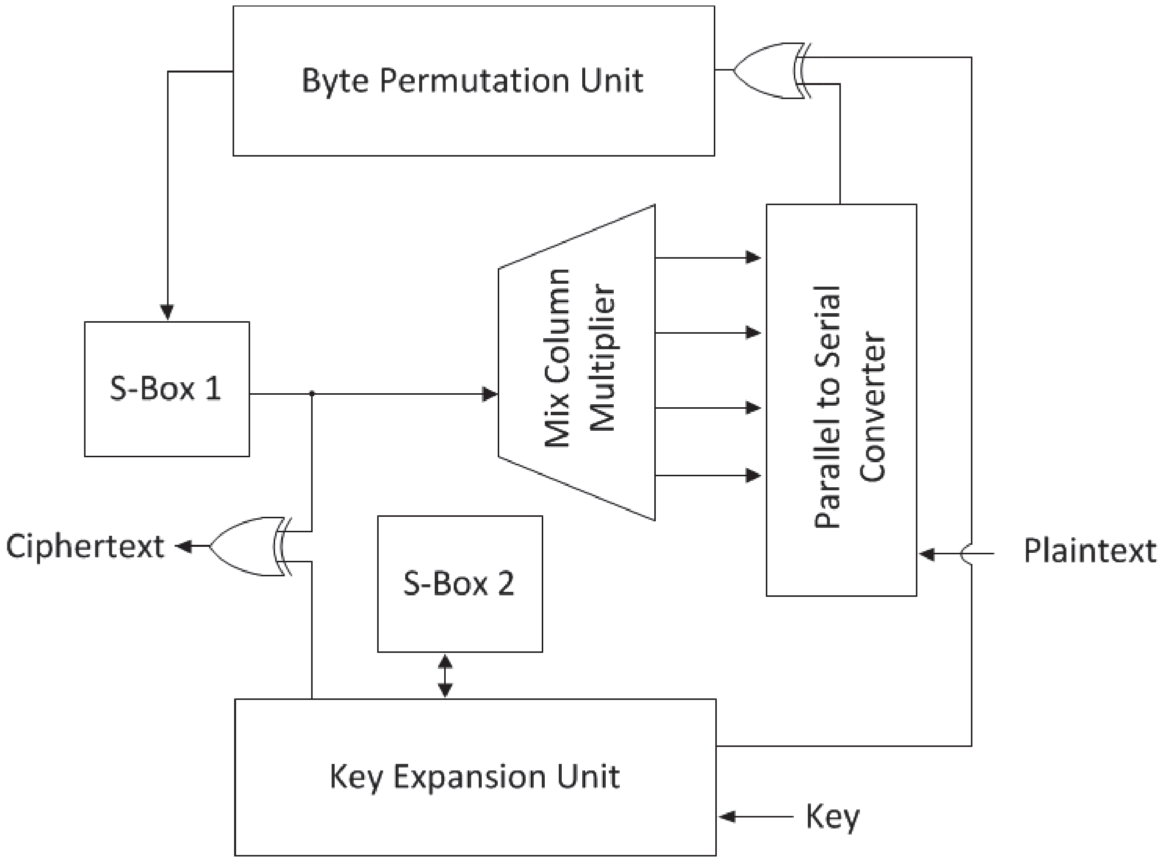


Figure 5.6 AES data path design

The design shown in Figure 5.6 was coded using VHDL. It was verified for functional correctness using Modelsim hardware simulation and verification tool. Appendix A shows a sample run of the design. The design was synthesized for Spartan 6 low power FPGA. Synthesis results are shown in Table 5.2. This design consumed about 3% of the device resources and it runs on 98.373 MHz.

Table 5.2 AES synthesis results

Parameter	**Value**
Number of Slice Registers	196
Number of Slice LUTs	338
Number of LUT Flip Flop pairs (LUT-FF)	369
Number of LUT-FF with an unused Flip Flop	173
Number of LUT-FF with an unused LUT	31
Number of fully used LUT-FF pairs	165
Maximum Frequency	98.373MHz
Device Utilization	3%

As will be explained in section 5.3.4, merging the AES algorithms with other designs on the final version of the SN-Sec platform will reduce the running frequency to about 58MHz. The final design power consumption will be 28 mJ. Using the number of cycles needed to run the AES algorithm, the AES running time can be computed as follows:

$$AES\ running\ time = \frac{Number\ of\ cycles\ needed\ to\ run\ the\ AES\ algorithm}{Running\ frequency\ of\ the\ SN-SEC} \quad (5.1)$$

$$= \frac{160\ cycles}{58\ MHz}$$

$$= 2.76\mu s$$

The energy consumption of SN-Sec processor for running the AES will be:

$$AES\ energy\ consuption = AES\ running\ time * SN-SEC\ power\ consumption \quad (5.2)$$

$$= 2.76\mu s * 28mJ$$

$$= 77.28\ nJ$$

These values outperform the results of related implementations discussed in Table 3.3. Running the AES hardware does not prevent the SN-Sec processor from performing other operations during encryption. In addition, the energy consumption results provided are for all components of the SN-Sec including AES.

5.3.2 Secure Hashing Algorithm SHA-256

As mentioned in section 3.4.2, SHA-256 is the most suitable secure hashing algorithm that meets the security requirements as well as the energy constraints of WSNs. Figure 5.7 shows the SHA-256 iteration as described in the FIPS standard [127].

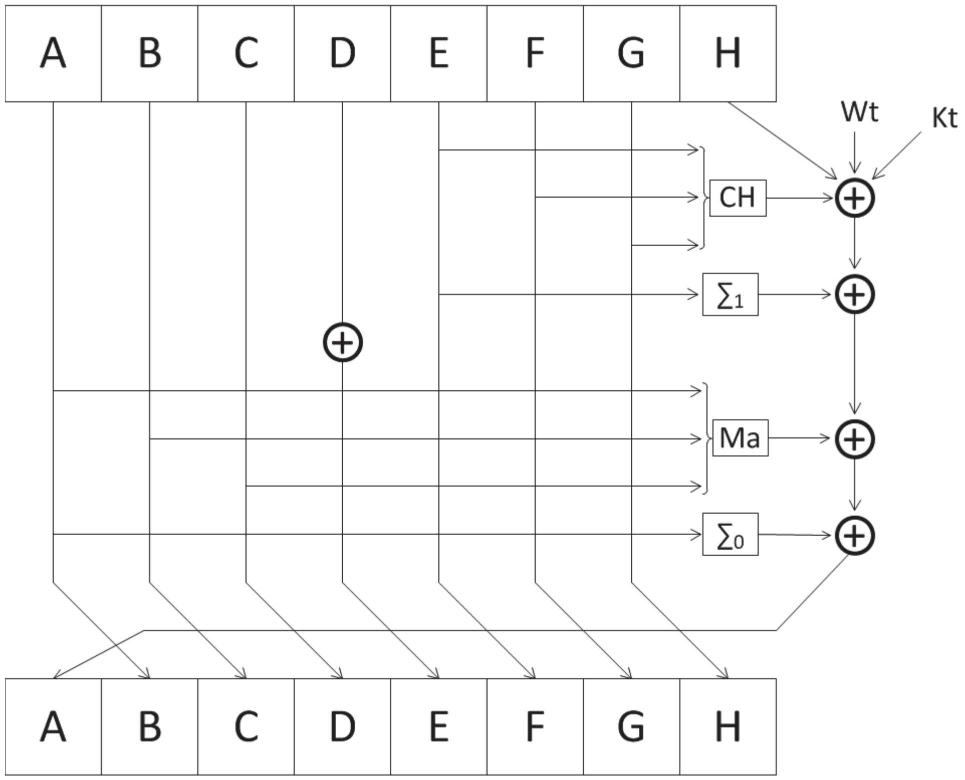


Figure 5.7 SHA-256 Iteration

Eight 32-bit constants are used to initialize the first iteration (i.e. A - F). SHA-256 has 64 iterations, and it uses 64 different 32-bit constants K_t for each iteration. On the other hand, SHA-384 and SHA-512 have 80 iterations and they use 80 different 64-bit constants for each iteration. SHA-3 operates on 1600-bit operands for 94 rounds. The larger operand size and number of constants requires more memory resources, and leads to higher energy consumption per block compared to SHA-256 [113]. According to the comparison between hardware synthesis results of SHA algorithms presented in section 3.4.2, SHA-256 has the lowest per block of energy and latency. This makes it the best choice for SN-Sec platform.

As for the design of the AES algorithm, the design in Figure 5.8 was coded using VHDL. It was verified for functional correctness using Modelsim hardware simulation and verification tool. Appendix A show a sample run of the design. The design was synthesized for Spartan 6 low power FPGA. Synthesis results are shown in Table 5.3.

Table 5.3 Synthesis results for SHA-256

Parameter	**Value**
Number of Slice Registers	1071
Number of Slice LUTs	1891
Number of LUT Flip Flop pairs (LUT-FF)	2005
Number of LUT-FF with an unused Flip Flop	934
Number of LUT-FF with an unused LUT	114
Number of fully used LUT-FF pairs	957
Maximum Frequency	59.977MHz
Device utilization	15%

The SHA-256 design consumes about 15% of the device resources and it runs on 59.977MHz. These values are higher than AES results because the SHA-256 design runs on 32-bit data-path compared to 8-bit data-path for AES. It also requires more hardware area to store and manipulate constants. This design requires 64 cycles to produce the message digest. Applying Equation 5.1 and 5.2 for SHA-256 shows that it takes 1.1 µs and 30.9 nJ to run the SHA-256 on the final design of the SN-Sec.

5.3.3 Elliptic Curve Processor

Selecting the curve parameters is a very important factor in the efficiency of the elliptic curve processor. The proposed elliptic curve processor operates in the 193 binary field because the security of lower size fields is not guaranteed to last long against brute force attack, as explained in section 3.4.3. The binary extension field was chosen because it requires less hardware area and energy consumption compared to the prime field.

Another important parameter used to reduce the computations of the proposed design compared to other designs proposed for WSNs is the choice of elliptic curve and associated coordinate system. A very efficient family of curves called Montgomery curves are used. They can be represented as in Equation 5.3:

$$by^2 = x^3 + ax^2 + x \qquad b(a^2 - 4) \neq 0 \tag{5.3}$$

A point $P = (x, y)$ on the Montgomery curve can be represented in the Montgomery projective coordinates without Y as $P = (X: Z)$ satisfying that $X = X/Z$ for $Z \neq 0$ [79]. Given two points in the Montgomery projective coordinates $P_\alpha = (X_\alpha: Z_\alpha)$ and $P_\beta = (X_\beta: Z_\beta)$, the sum of the two points can be defined $P_{\alpha+\beta} = (X_{\alpha+\beta}: Z_{\alpha+\beta})$ where:

$$X_{\alpha+\beta} = Z_{\alpha-\beta}\left((X_\alpha - Z_\alpha)(X_\beta + Z_\beta) + (X_\alpha + Z_\alpha)(X_\beta - Z_\beta)\right)^2 \tag{5.4}$$

$$Z_{\alpha+\beta} = X_{\alpha-\beta}\left((X_\alpha - Z_\alpha)(X_\beta + Z_\beta) + (X_\alpha + Z_\alpha)(X_\beta - Z_\beta)\right)^2 \tag{5.5}$$

This operation has a time-cost of four modular multiplications and two modular squaring's, assuming that $Z_1 = 1$. The cost of point addition will be reduced to three modular multiplications. If $\alpha = \beta$, the outcome becomes point doubling $\mathrm{P}_{2\alpha}(\mathrm{X}_{2\alpha}: \mathrm{Z}_{2\alpha})$ which can be expressed as:

$$4X_\alpha Z_\alpha = (X_\alpha + Z_\alpha)^2 - (X_\alpha - Z_\alpha)^2$$

$$X_{2\alpha} = (X_\alpha + Z_\alpha)^2 (X_\alpha - Z_\alpha)^2 \tag{5.6}$$

$$Z_{2\alpha} = 4X_\alpha Z_\alpha\left((X_\alpha - Z_\alpha)^2 + \left((a + 2)/4\right)4X_\alpha Z_\alpha\right) \tag{5.7}$$

The cost of multiplying by a in Equation 5.5 can be ignored if the value of a is small, hence the doubling operation will have a cost of two modular multiplications and two modular squarings [128]. This is the most efficient coordinate system up to our knowledge; it has fewer operations for point doubling and point addition than all the coordinate systems used for related work in Table 3.9.

Unlike the Double-and-Add scalar point multiplication in Algorithm 3.1, the Montgomery ladder algorithm [79] computes the scalar point multiplication in a fixed amount of time and constant power consumption. This can be beneficial in WSNs environments, where timing and power consumption of the sensor processor are exposed to any adversary performing a side channel attack. The Montgomery ladder algorithm does not leak any information through timing or power. Algorithm 5.2 shows the Montgomery ladder algorithm.

Algorithm 5.1 Montgomery Ladder point multiplication

```
$\boldsymbol{Input}: P: EC\ Point, Scalar\ n > 0,\ n: n_{l-1}, \cdots, n_1, n_0.$
$\boldsymbol{Output}: Q = nP$

  $Q_0 \leftarrow P, Q_1 \leftarrow 2P$ ;
  $\boldsymbol{for}\ i = l-2\ down\ to\ 0$
      $\boldsymbol{if}\ (n_i = 0)$
          $Q_1 = Q_0 + Q_1,\ Q_0 = 2Q_0$;
      $\boldsymbol{else}$
          $Q_0 = Q_0 + Q_1,\ Q_1 = 2Q_1$;
  $\boldsymbol{endif}$
  $\boldsymbol{return}\ Q_0$;
```

However, the Montgomery ladder algorithm will increase the number of arithmetic operations because a point addition as well as a point doubling has to be accomplished at each iteration of the algorithm. As shown earlier in this section, using the Montgomery curve in the XZ coordinates point doubling is accomplished with two multiplications and two squarings and point addition with three multiplications and two squarings. The total is five multiplications and four squarings. Table 5.4 shows a detailed description of elliptic curve operations in one iteration of the Montgomery ladder algorithm based on Montgomery curve in the XZ coordinate. The operations start with the coordinates of the input points $(X_{\alpha}: Z_{\alpha})$ and $(X_{\beta}: Z_{\beta})$. It applies Equation 5.4 and 5.5 to produce the result of point addition $(X_{add}: Z_{add})$, and point doubling $(X_{double}: Z_{double})$.

Table 5.4 Elliptic curve operations in one iteration of the Montgomery ladder algorithm in the XZ coordinate

Operation	**Type**	**Dependency**
$A \leftarrow X_{\alpha} + Z_{\alpha}$	A_0	-
$AA \leftarrow A^2$	S_0	A_0
$B \leftarrow X_{\alpha} - Z_{\alpha}$	A_1	-
$BB \leftarrow B^2$	S_1	A_1
$E \leftarrow AA - BB$	A_2	A_1 A_0
$C \leftarrow X_{\beta} + Z_{\beta}$	A_3	-
$D \leftarrow X_{\beta} - Z_{\beta}$	A_4	-
$DA \leftarrow D * A$	M_0	A_4A_0
$CB \leftarrow C * B$	M_1	A_3A_1
$X_{add} \leftarrow (DA + CB)^2$	A_5+ S_2	M_0M_1
$Z_{add} \leftarrow X_{\alpha} * (DA - CB)^2$	A_6+ M_2+ S_3	M_0M_1
$X_{double} \leftarrow AA * BB$	M_3	S_0,S_1
$Z_{double} \leftarrow E * (BB + E * (a - 2)/4)$	M_4+A_7+C_0	S_0S_1
Total	7A+4S+5M+C	
Critical path	2A+5M	

Figure 5.9 shows the critical path of the elliptic curve operations in Table 5.4. Assuming that modular addition, squaring, and multiplication hardware is independent and can run in parallel, and that modular multiplication in more costly than modular squaring, it can be shown that one iteration of the Montgomery ladder algorithm can be implemented within the critical path of five multiplications, and two additions i.e. A_4A_0 before M_0. Hence, the total cost for elliptic curve point multiplication will be: $I + n(5M + 2A)$. This is the best value compared to the elliptic curve hardware implementations proposed for WSNs presented in Table 3.9.

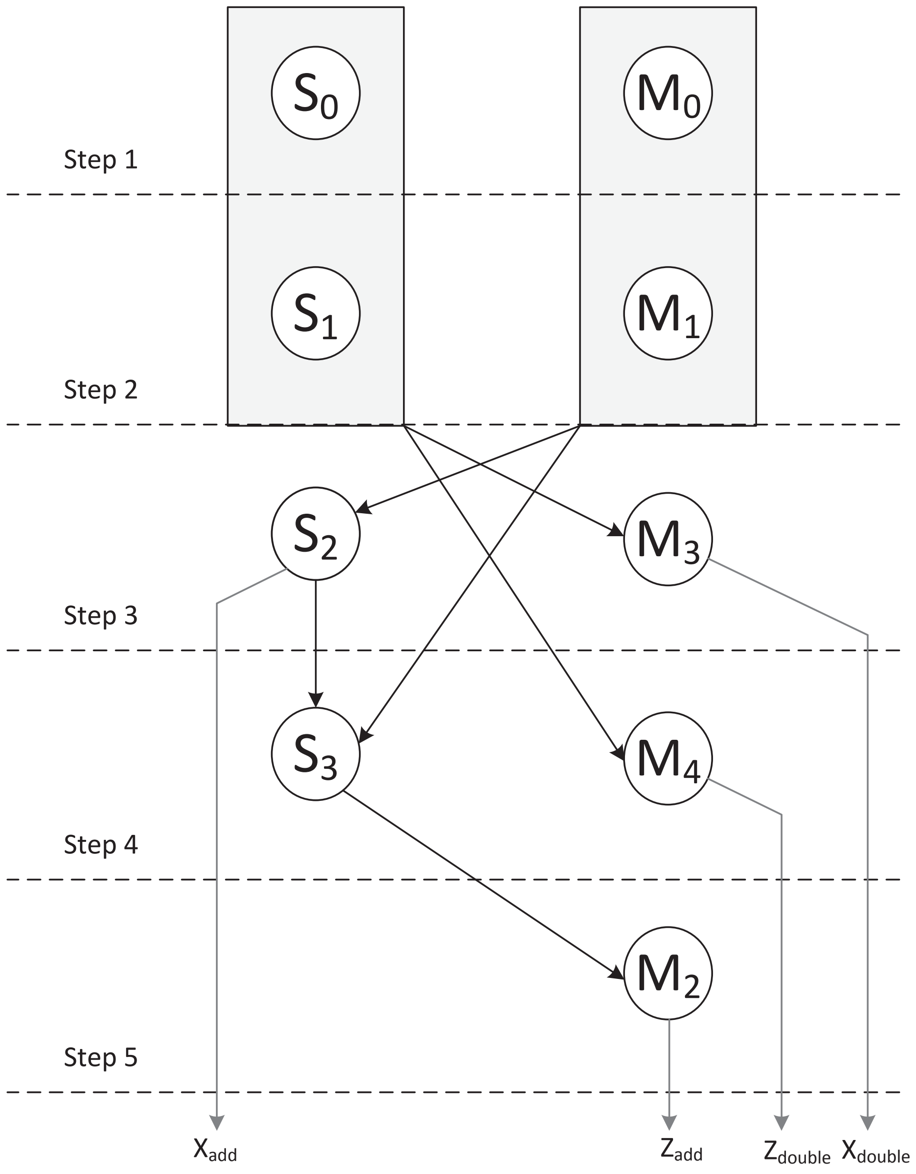


Figure 5.8 Dependency between elliptic curve operations

Efficient implementation of multiplication squaring, and inversion algorithms are the next important factors that play a role in the efficiency of the elliptic curve processor. Redundant interleaved modular multiplier is used for modular multiplication for the SN-Sec. Algorithm 5.3 shows the interleaved modular multiplication algorithm. The multiplication and reduction steps overlap in each iteration of the algorithm. The intermediate results are reduced after each iteration before resuming the multiplication process in the next iteration.

Algorithm 5.2 Interleaved modular multiplication

```
$\boldsymbol{Input}: X, Y, M;\ n\ bits\ numbers;\ 0 \le X,\ \ Y \le M;\ x_i: i^{th}\ bit\ of X$
$\boldsymbol{Output}: X \times Y\ mod\ M$

 $P = 0;$
$\boldsymbol{for}\ i = n - 1\ down\ to\ 0$
    $P = 2 \times P;$
    $I = x_i \times Y;$
    $P = P + I;$
    $\boldsymbol{if}(P \ge M)\ P = P - M;\ \boldsymbol{endif}$
    $\boldsymbol{if}(P \ge M)\ P = P - M;\ \boldsymbol{endif}$
$\boldsymbol{endfor}$
$\boldsymbol{return}\ P;$
```

The efficiency of interleaved multiplication can be improved by pre-estimating the number of times M has to be subtracted at the last two steps of the algorithm, and predicting if Y has to be added in the next iteration of the loop. All these choices can be precomputed and stored in a lookup table. In each iteration of the loop, the estimation from the previous step is added to the intermediate result. With the use of carry save addition the complexity of computing multiplication can be reduced to n cycles. Figure 5.10 shows the architecture of the interleaved modular multiplier, a more detailed description of this architecture can be found in [129].

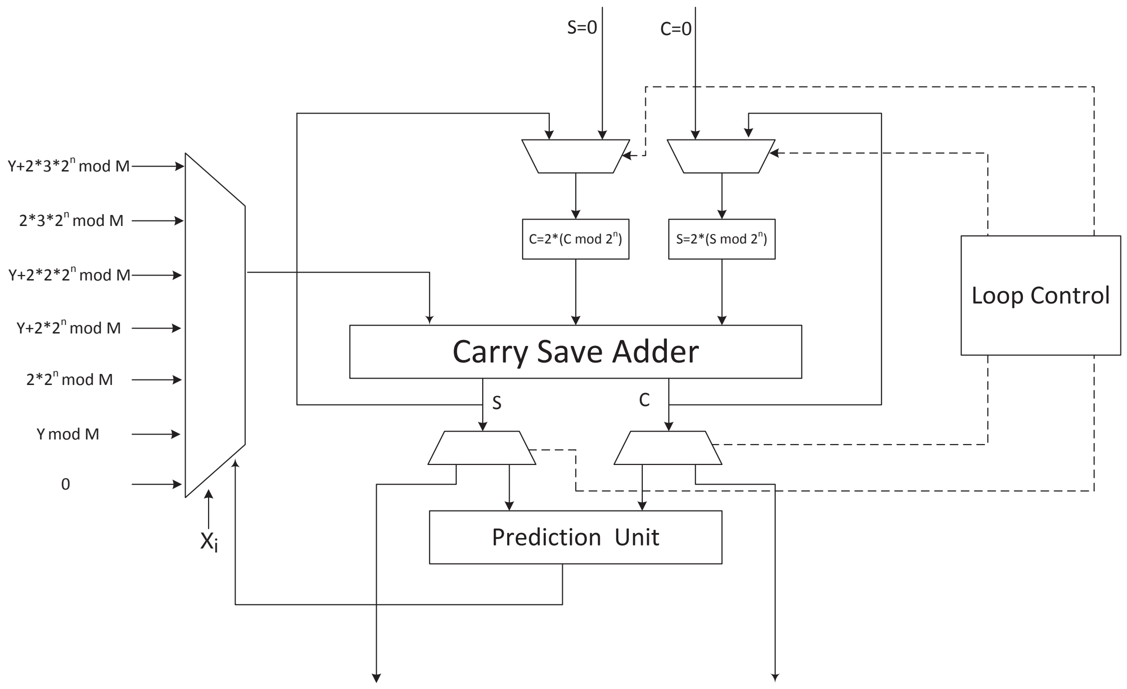


Figure 5.9 Interleaved Modular Multiplier

The computation of the modular squaring can be optimized compared to modular multiplication due to the fact that cross terms of the intermediate results disappear because they come in pairs and the underlying field is the binary field. Using the polynomial representation for the elements of the $GF(2^n)$ field, an element a of the field is a polynomial of length n, i.e. of degrees less than or equal $n-1$, can be written as:

$$\begin{aligned} a(x) &= \sum_{i=0}^{n-1} a_i x^i \\ &= a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_2x^2 + a_1x + a_0. \end{aligned} \qquad (5.8)$$

Where the coefficients $a_i \in GF(2)$. These coefficients can be referred to as the bits of a and the elements of a can be represented as

$$a = a_{n-1}, a_{n-2}, \cdots, a_1, a_0. \qquad (5.9)$$

Hence, it is easy to show that the squaring of the element a will be:

$$a(x)^2 = \sum_{i=0}^{n-1} a_i x_i^{2i} \tag{5.10}$$

The Montgomery squaring algorithm involves a polynomial $c(x) = a(x)^2$ with degree $2(n-1)$:

$$c(x) = a_{n-1}x_{n-1}^{2(n-1)} + a_{n-2}x_{n-2}^{2(n-2)} + \cdots + a_1x^2 + a_0. \tag{5.11}$$

$$= a_{n-1}, 0, a_{n-2}, 0, \cdots, 0, a_1, 0, a_0. \tag{5.12}$$

Then $c(x)$ can be reduced by computing $c(x) = c(x)x^{-n} mod\ M.$ Algorithm 5.4 explains the steps of this procedure.

Algorithm 5.4 Montgomery modular squaring algorithm

```
$\textbf{Input}: X, M;\ n\ bits\ numbers;\ 0 \le X;\ x_i: i^{th}\ bit\ of X$
$\textbf{Output}: C = c(x)^2\ x^{-n} mod\ M$

    $C = \sum_{i=0}^{n-1} a_i x_i^{2i}$
    $\textbf{for}\ i = 0\ to\ n-1$
        $C = C + c_0 * M;$
        $C = C/2;$
    $\textbf{endfor}$
    $\textbf{return}\ C;$
```

The term c_0 is the least significant bit of C. Its value is zero half of the time. The only operation done in the loop when c_0 equals zero is shift right. Pre-computing c_0, then performing the loop only when its value is one, with the amount of shifts done on C equals to the amount of skipped zeros in c_0. This will save half of the iterations on average. Hence, the cost of squaring will be $n/2$ cycles on average. More about this algorithm can be found in [120] [131].

Converting back to the affine coordinate system is necessary to produce the final elliptic curve point multiplication results. This conversion requires one final modular division operation. The extended Binary Greatest Common Devisor (BGCD) algorithm [122] is a very efficient way to compute modular division. Algorithm 5.4 shows the detailed steps of this algorithm.

Algorithm 5.3 Modular Division in $GF(2^m)$

```
$\boldsymbol{Input}$: $X, Y, and\ M$ : $numbers\ in\ GF(2^n);\ Y \neq 0.$
$\boldsymbol{Output}$: $V = X/Y\ mod\ M$
$\boldsymbol{Initialize}$: $U = X, R = Y, S = G = M, V = 0, count = 0, state = 0.$

    $\boldsymbol{for}\ i = 1\ \ to\ 2n\ \boldsymbol{do}$
        $\boldsymbol{if}(state = 0)$
            $count = count + 1;$
            $\boldsymbol{if}(r_0 = 1)$
                $(R, S) = (R + S, R);$
                $(U, V) = (U + V, U);$
                $state = 1;$
            $\boldsymbol{endif}$
        $\boldsymbol{else}$
            $count = count - 1;$
            $\boldsymbol{if}(r_0 = 1)$
                $(R, S) = (R + S, S);$
                $(U, V) = (U + V, V);$
            $\boldsymbol{endif}$
            $\boldsymbol{if}(count = 0)$
                $state = 0;$
            $\boldsymbol{endif}$
        $\boldsymbol{endif}$
        $R = R/2;$
        $U = U/2;$
    $\boldsymbol{endfor}$
```

The division is performed by interleaving the procedure for finding the modular quotient with that for calculating the greatest common divisor (GCD) of two numbers in the binary extended field. More about this algorithm can be found in [133]. It is mainly based on three rules:

- $GCD(R, S) = 2 * GCD(S/2, R/2)$, If R and S are even
- $GCD(R, S) = GCD(S, R/2)$, If R is even and S is odd
- $GCD(R, S) = GCD((S - R)/2, R)$, If R and S are odd

Using only one redundant modular adder for the design; knowing that addition takes one clock cycle, swapping each operand takes one clock cycle, and that all other shift, increment and decrement operations are done on the fly, this algorithm will take $8n$ cycles.

The elliptic-curve point multiplication can be computed in $I + n(5M + 2A)$ cycles using this design. Knowing that modular inversion I takes $8n$ cycles, modular addition A takes one cycle, and modular multiplication M takes 193 cycles, and the field size n is 193. The total number of cycles to compute one point multiplication is:

$$= 8 * 193 + 193(5 * 193 + 2)$$

$$= 188175 \text{ Cycles.}$$

The critical path of the proposed elliptic curve processor is the most efficient among related work because it includes the least amount of computations, the most efficient components designs, and it exploits parallelism between components to produce the result in small number of cycles. As Figure 5.11 shows, the elliptic curve processor has four modular components that can be run in parallel by the control unit.

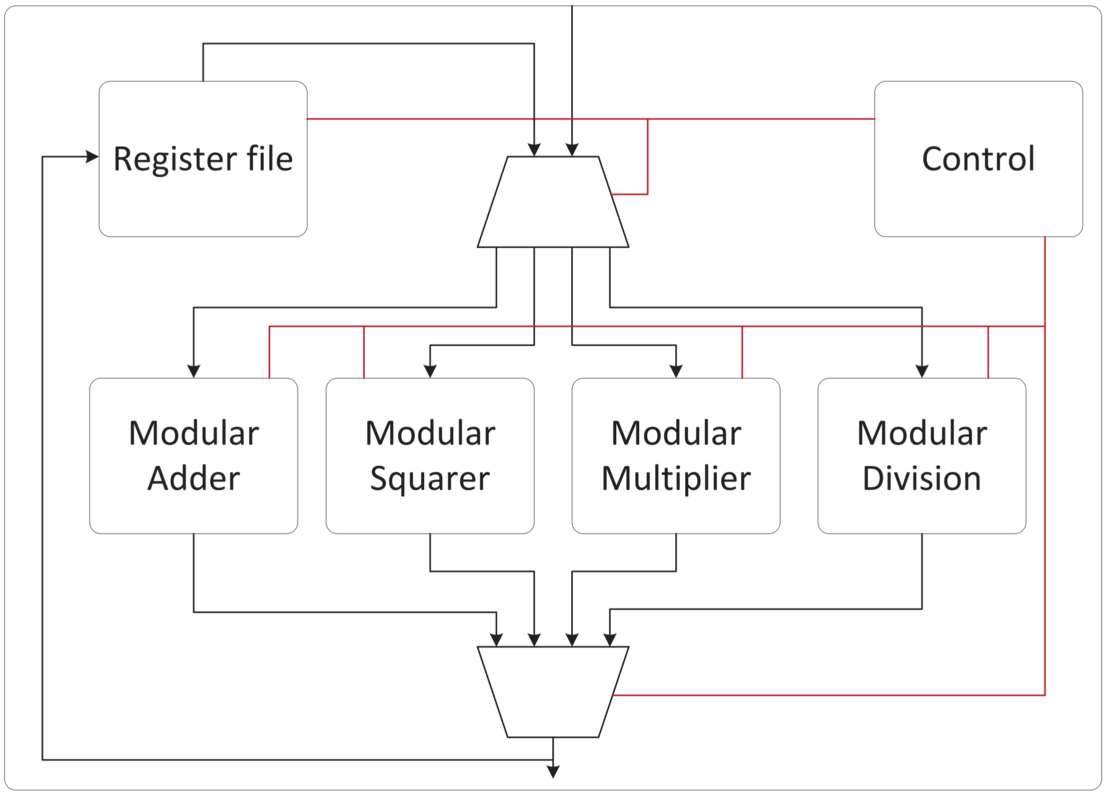


Figure 5.10 ECC design

This design was coded in VHDL and verified for functional correctness using Modelsim hardware simulation and verification tool. Appendix A shows a sample run of this algorithm. Table 5.5 shows the synthesis results for the elliptic curve processor for the Spartan 6 low power FPGA. It consumes 60% of the device resources and operates on 57.998MHz. As will be shown in the next section the SN-Sec operates at that frequency, which indicates that the elliptic curve processor includes the longest critical path of the SN-Sec. Applying Equation 5.1 and 5.2 for the elliptic curve processor show that it requires 3.245 ms and consumes 90.86 µJ to compute one elliptic curve point multiplication on the final SN-Sec processor.

Table 5.5 Synthesis results for elliptic curve processor

Parameter	Value
Number of Slice Registers	2805
Number of Slice LUTs	5518
Number of LUT Flip Flop pairs (LUT-FF)	6132
Number of LUT-FF with an unused Flip Flop	3327
Number of LUT-FF with an unused LUT	614
Number of fully used LUT-FF pairs	2191
Maximum Frequency	57.998MHz
Device utilization	60%

5.3.4 THE SN-Sec

Figure 5.12 shows the design of the SN-Sec platform. It consists of three main parts, a 32-bit Reduced Instruction Set Computing (RISC) processor; an encryption processor that contains the hardware implementations of AES, ECC and SHA with its own 32 kB dual port memory for secure key storage that is not readable by the processor; and a 512 kB internal memory module readable by the processor to store the C-Sec caches and encryption primitives input/output.

The ZPU processor is a modified version of the 32-bit RISC open source Zylin Processor Unit (ZPU) [25]. New instructions were added to the Instruction Set Architecture (ISA) of the ZPU, these instructions are shown in Table 5.6. The new instructions controls each of the crypto-primitives implemented in the encryption processor and to provide the input plaintext, location of encryption keys and the addresses where the output should be placed. This is done by writing to specific locations in the dual port memory of the encryption processor, the main processor can only write to that memory using the write only port of the dual port memory.

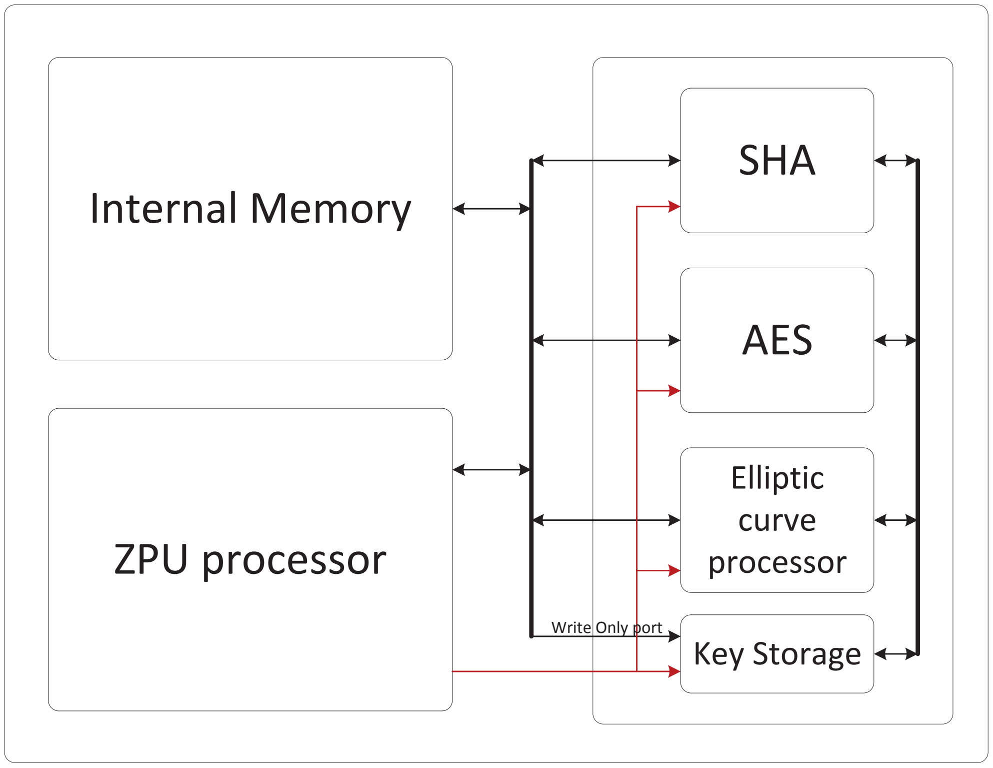


Figure 5.11 SN-Sec architecture

For each of the instructions that activate an encryption primitive, the location of the key is stored in the secure-memory address register *mReg,* the address of input location is stored in the input register *iReg,* and the address of the memory where the output has to be written is stored in the output register *oReg*. The ***Mem_w*** instruction is used to write the value in the address specified in the input register *iReg* to the secure-key storage memory location stored in the *mReg*.

Table 5.6 New instructions added to the ZPU ISA

Instruction	Op-code	Description
SHA_en	1001 0000	Activate SHA Algorithm. Key is stored at location in *mReg* of the secure key storage. Input location is in *iReg* of the internal memory, write output to address in *oReg* of the internal memory.
AES_en	1010 0001	Activate AES in encryption mode. Key is stored at location in *mReg* of the secure key storage. Input location is in *iReg* of the internal memory, write output to address in *oReg* of the internal memory.
AES_de	1010 0010	Activate AES in encryption mode. Key is stored at location in *mReg* of the secure key storage. Input location is in *iReg* of the internal memory, write output to address in *oReg* of the internal memory.
ECC_en	1011 0011	Activate ECC encryption. Public key is stored at location in *mReg* of the secure key storage. Input location is in *iReg* of the internal memory, write output to address in *oReg* of the internal memory.
ECC_de	1011 0100	Activate ECC decryption. Private key is stored at location in *mReg* of the secure key storage. Input location is in *iReg* of the internal memory, write output to address in *oReg* of the internal memory.
Mem_w	1100 0000	Write value in address stored in *iReg* of the internal memory to the memory address to the *mReg* of the secure key storage

The secure key storage memory is configured to be accessed by the ZPU in the write mode only. This is done by keeping the read/write input of the memory port connected to the ZPU processor on the write mode by assigning the *we_0 <='1'*, and keeping the output enable port disabled by assigning *oe_0<='0',* as shown in Figure 5.13. It is assumed that the main processor is programmed to delete the keys after they are loaded to that memory. This makes it difficult for the attacker to get the keys back if she manages to take control of the OS, because physically she cannot read them back from the dual port memory using any program running on the OS. This will help secure the key storage and reduce the effect of compromising sensor nodes. The output of the encryption processor can be written to specific locations on the internal memory.

```
DPMMAP: ram_dp_ar_aw
      port map(
          address_0 =>Add0,   --address
          data_0    =>d_0,    -- data
          cs_0      =>cs_0,   -- Chip Select
          we_0      =>'1',    -- Write Enable/Read Enable
          oe_0      =>'0',    -- Output Enable
          address_1 =>Add1s,
          data_1    =>d_1s,
          cs_1      =>cs_1s,
          we_1      =>we_1s,
          oe_1      =>oe_1s
     );
```

Figure 5.12 Configuration of the Dual port memory for the secure key storage

The modified ZPU design was synthesized for Spartan 6 low power FPGA. Synthesis results are shown in Table 5.7. This design consumed less than 1% of the device resources and it runs on 227.095MHz.

Table 5.7 synthesis results for the ZPU

Parameter	**Value**
Number of Slice Registers	23
Number of Slice LUTs	30
Number of LUT Flip Flop pairs (LUT-FF)	34
Number of LUT-FF with an unused Flip Flop	21
Number of LUT-FF with an unused LUT	5
Number of fully used LUT-FF pairs	8
Maximum Frequency	227.095MHz
Device utilization	1%

The components of the SN-Sec platform were integrated together. The synthesis results for the whole design for the Spartan 6 xc6slx16 FPGA are shown in Table 5.8. The design consumed 71% of the device resources. The internal memory consumed all the RAM blocks of the device. The power results obtained from the Xilinx X-power analyzer tool show that the device consumes 28 mJ.

Table 5.8 Synthesis results for SN-Sec

Parameter	**Value**
Number of Slice Registers	4272
Number of Slice LUTs	7371
Number of LUT Flip Flop pairs (LUT-FF)	8715
Number of LUT-FF with an unused Flip Flop	4810
Number of LUT-FF with an unused LUT	519
Number of fully used LUT-FF pairs	3386
Block RAM/FIFO	32
Maximum Frequency	57.998MHz
Device utilization	71%

Table 5.9 shows the detailed results for the number of cycles, running time, and energy consumption for each of the SN-Sec components. Figure 5.14 shows the approximate breakdown of the device resources consumed by each of these components. Note that the elliptic curve design consumes more than 60% of the device resources; on the other hand the ZPU processing unit consumes only 1%. This gives an indication of the complexity and large design of the elliptic curve processor architecture.

Table 5.9: Hardware implementations results

Component	*Frequency (MHz)*	*Power (mw)*	*Clock Cycles*	*Time (µs)*	*Energy (nJ)*
AES-128	57.998	28	160	2.76	77.28
SHA-256			64	1.1	30.9
ECC-193			188175	3245	90860

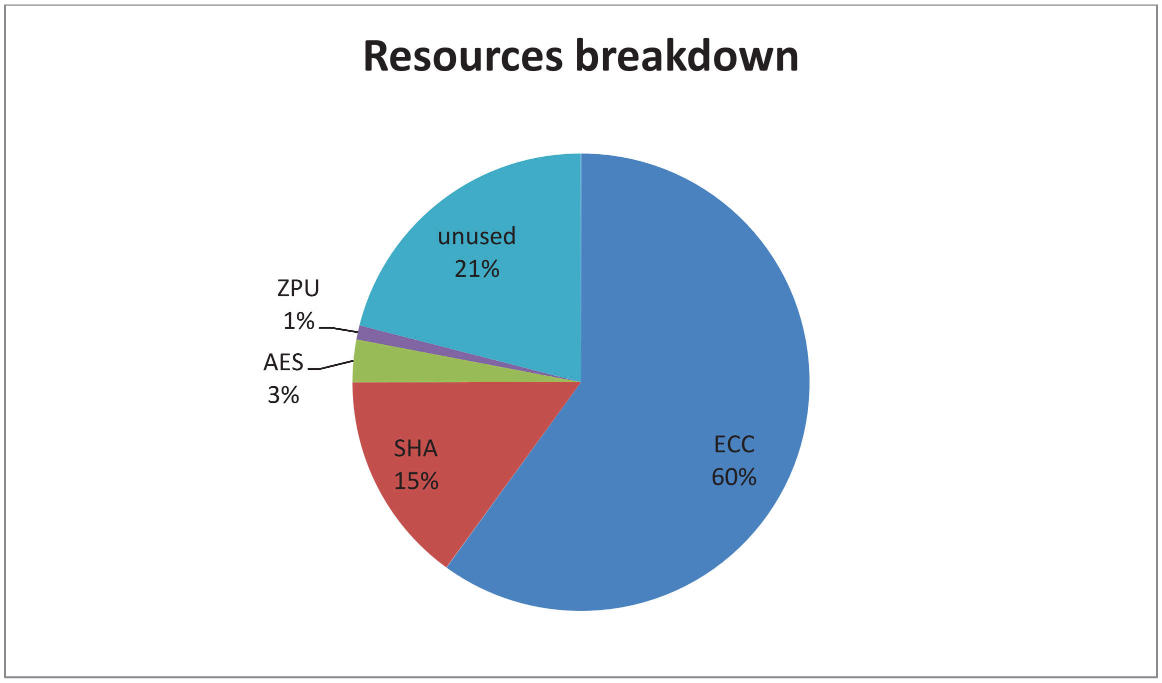


Figure 5.13 Approximate device utilization breakdown

Table 5.10 shows a list of well-known WSN processors. The SN-Sec outperforms these processors in terms of the achieved level of security. In addition, the SN-Sec platform implements complex encryption primitives in hardware that consume about 78% of the device resources as shown in Figure 5.14. However, it has a reasonable power consumption compared to these processors. This allows implementing highly secure wireless nodes that run security protocols based on complex cryptographic primitives in short periods of time, without affecting the execution of other programs running on the sensor node processor, and without significantly affecting the life time of the wireless sensor node.

Table 5.10 Comparison with other WSNs processors

Processor	*Power (mw)*	*Data-path width (bit)*	*Memory (kB)*	*Frequency (MHz)*	*Energy (pJ/Ins)*	*Technology (nm)*	*Hardware Encryption*
SN-Sec	28	32	512	57.998	483	FPGA	yes
Intel StrongARM	400	32	16	206	1942	350	No
Texas Instruments MSP 430	6	16	10	8	750	350	No
BitSNAP [124]	43.6	16	8	200	218	180	No
[125]	1.78	16	128	0.434	27	65	No
Atmel ATmega 128L	75	8	4	16	3200	350	No
[126]	0.768	8	68	8	96	130	No

5.4 CONCLUDING REMARKS

Chapter 5 emphasized the security and energy efficiency of the SN-Sec, and demonstrated the advantages of having energy efficient implementation of encryption primitives in hardware, especially the elliptic curve processor included in the SN-Sec implementation. The SN-Sec platform improves resilience against node compromise. If a node's operating system is compromised, then the key storage will not be accessible. However, if a node that uses other architectures is compromised, then key materials inside the node including shared keys with other nodes will be exposed.

In addition, SN-Sec provides resilience against board level attacks like bus probing and on-board memory attacks. SN-Sec is a system on chip solution. All communication between critical components of the system are done inside the scope of the chip, which eliminates the chance of bus probing attacks. All other communication outside the chip scope is encrypted; the onboard memory is encrypted as well, which eliminates the possibility of onboard memory attacks. The cost of encrypting this memory is dramatically reduced compared to other platforms because of the energy and time efficient hardware implementation of the AES algorithm in SN-Sec, and because running encryption primitives does not interrupt or block the operation of the main processor, since the algorithm run on separate independent hardware.

The crypto-primitives designs proposed for SN-Sec are the most efficient compared to other designs proposed for WSNs up to our knowledge. The implementation of AES algorithm combines the most energy efficient data-path [70] and S-Box design [66] in the literature. The SHA-256 implementation used has the least per block energy consumption and latency compared to other SHA family algorithms. This makes it the best choice for WSNs. The proposed elliptic curve processor has the least amount of modular operations among all elliptic curve processors proposed for WSNs up to our knowledge. Its design exploits parallelism among these operations to reduce the number cycles needed to perform elliptic curve point multiplication, with a design immune to power analysis and timing attacks.

The SN-Sec is implemented on an FPGA, which helps to resist tampering attacks. Implementing the processing unit and other necessary hardware designs on an FPGA instead of microcontroller-based devices makes it much more expensive for reverse engineering. “Even placing important algorithms inside a ‘non-secure’ SRAM-based FPGA makes its reverse engineering very difficult” [22].

Key management protocols handle the task of distributing, establishing, and managing keys between sensor nodes. The SN-Sec platform facilitates implementing asymmetric based key management protocols, which have higher security strength, scalability and connectivity over other key management mechanisms that do not use asymmetric encryption primitives. It also has low storage and communication complexity, high scalability and resilience to node compromise [105]. The only drawback of asymmetric based key management protocols is the computational complexity, which leads to unacceptable energy and latency requirements; this issue has been resolved with the energy and latency efficient hardware design of the SN-Sec.

The energy efficient asymmetric encryption provided by the SN-Sec facilitates implementing many security services and mechanisms for WSNs, like providing the non-reputation security service through certification, public key infrastructures [137], broadcast authentication [138], ID-based key agreement schemes [105], and asymmetric based homomorphic encryption [139]. These services were considered infeasible due to the computational complexity of asymmetric encryption.

CHAPTER 6 CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This thesis presented solutions to security and its related energy efficiency in WSNs. This was addressed at the link layer by proposing C-Sec, a dual mode encryption protocol that reduces the energy cost of communication with a small increase in processing, which has much lower-and relatively decreasing energy cost. This is done by excluding the requirement for explicitly transmitting all header fields related to security most of the time. An evaluation of the security services C-Sec can provide shows that the C-Sec does include all the security services provided by other protocols. In addition, it hides the packet header in the compact mode, which makes it more difficult to eavesdrop on the flow of wireless communication. It also reduces the effect of message spoofing and replay attacks and works as an attack detection mechanism.

An evaluation of C-Sec using the Castalia simulator shows that it saves more energy than all related protocols and the TinyOS under different packet loads and sizes. The saving in energy can reach up to 19.1% compared to MiniSec, the encryption protocol with closest packet size to C-Sec. Both of the mathematical analysis of the C-Sec queuing model, and the simulation results of the additional overhead time introduced by C-Sec, show that as the traffic load increases, the additional waiting time introduced by the C-Sec queuing model decreases. They also show that C-Sec time overhead increases with lower traffic loads. However, this increase is capped, because packets with waiting times larger than the authentication timer are transmitted in the conventional mode. The additional packet delay overhead of C-Sec can be tolerated, as most of WSNs applications do not require hard real-time constraints. The significance of the energy savings the C-Sec can achieve outweighs the additional packet delay it introduces.

Mathematical analysis of the post-decryption error probability of the C-Sec shows that the C-Sec has slightly increases the error probability. Compare to MiniSec, the protocol smallest packet size among related protocols and the closest to C-Sec, the difference in post-decryption error probability does not exceed 1% in the active region. Simulation results show that packet loss increases with the increase of noise floor. However, C-Sec has about 1% more dropped packets

WSN security and its related energy consumption was also addressed at the physical layer by proposing SN-Sec, a secure sensor platform that is based on a modified version of the ZPU processor by adding new instructions to operate hardware implemented encryption primitives. SN-Sec provides immunity against board level attacks and resilient to node compromise by having a secure key storage that is not accessible by the operating system. SN-Sec uses the most energy efficient hardware implementations of encryption primitives, to our knowledge. The implementation of the AES algorithm combines the most efficient data-path design with the most efficient S-box design in the literature. All the SHA family members were implemented in VHDL. A comparison between hardware areas, energy consumption per block, and latency results for hardware synthesis of their implementations was performed. It was found that SHA-256 has the least per-block energy consumption and latency. An energy efficient elliptic-curve processor design was proposed, it has the least number of operations, as well as, clock cycles per encryption among elliptic curve processors proposed for WSNs. The proposed processor exploits the parallelism in XZ coordinate to reduce number of cycles.

Each of the SN-Sec design components was implemented using VHDL, their functional correctness was tested using ModelSim, and they were synthesized for Spartan 6 low power FPGA using Xilinx design suite. This included providing hardware area, frequency, time delay, and energy consumption for each algorithm separately and for the integrated system.

6.2 FUTURE WORK

The following list contains a summary of several research topics that will be pursued in the near future as a continuation of research work presented in this thesis:

- Integrating a more comprehensive and energy efficient system on chip solution for WSNs, by adding more energy efficient hardware components needed for sensor node operation to the proposed SN-Sec platform, like adding an energy efficient transceiver circuit, an analog to digital converter, and any other digital signal processing components or computational intensive algorithms used in WSN environments.

- To implement the SN-Sec design using state of the art programmable devices, like the new Artix 7 [140] devices manufactured by Xilinx. This will result in significant power saving. The design cycle will be completed by downloading the generated bit stream of the design to the FPGA device using the Xilinx development board.

- To develop a secure sensor node that uses the SN-Sec platform, and then implement the C-Sec protocol on WSNs of these new nodes to conduct further evaluation for its functionality and efficiency, and provide a complete WSN solution that can be used in real life applications.

- The energy efficient asymmetric encryption provided by the SN-Sec facilitates implementing many security services and mechanisms for WSNs. These services were determined to be infeasible due to the computational complexity of asymmetric encryption. Continued research will be carried out in many directions in this area, such as:

- To develop a public key infrastructure [137] that defines energy an efficient mechanism for secret key distribution and provides non-repudiation service through certification.

- To design energy efficient mechanisms for broadcast and multicast authentication [138] for WSNs. Broadcast and multicast authentication are fundamental to realizing the security of user commands and on demand query functions, which are essential parts of the WSN operation.

- To develop new elliptic curve based key management schemes with high security levels and minimal computation and communication cost. This can be done after evaluating the security, communication overhead and computational complexity of existing key management schemes, which fall into three types: pure asymmetric schemes, hybrid schemes, and ID-based schemes.

- To develop elliptic curve based homomorphic encryption mechanisms [139] for concealed data aggregation. This is a very new field that is full of research potential. The elliptic curve implementation provided by SN-Sec can be optimised to support such mechanisms efficiently.

BIBLIOGRAPHY

[1] Tomas Sanchez, Damith Ranasinghe, Mark Harrison, Duncan McFarlane, "Adding sense to the Internet of Things - An architecture framework for Smart Object systems," *Personal and Ubiquitous Computing,* vol. 16, no. 3, pp. 291-308, 2012.

[2] Peter Harrop , "Wireless sensor networks 2009-2019," Technical Report, IDTechEx, November, 2008.

[3] Martin Floeck, Lothar Litz, "Activity- and Inactivity-Based Approaches to Analyze an Assisted Living Environment," in *Emerging Security Information, Systems and Technologies (SECURWARE)*, 2008.

[4] Venkata Kottapalli, Anne Kiremidjian, Jerome Lynch, Ed Carryer, Thomas Kenny, Kincho Law, Ying Lei, "Two-tiered wireless sensor network architecture for structural health monitoring," in *SPIE 5057, Smart Structures and Materials 2003: Smart Systems and Nondestructive Evaluation for Civil Infrastructures*, San Diego, 2003.

[5] Kay Soon Low, Win Nu Nu Win, Meng Joo Er, "Wireless Sensor Networks for Industrial Environments," in *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, 2005.

[6] Sang Hyuk Lee, Soobin Lee, Heecheol Song, Hwang Soo Lee, "Wireless sensor network design for tactical military applications : Remote large-scale environments," in *Military Communications Conference (MILCOM)*, 2009.

[7] Ali Maleki Tabar, Arezou Keshavarz, Hamid Aghajan, "Smart Home Care Network using Sensor Fusion and Distributed Vision-based Reasoning," in *4th acm international workshop on video surveillance and sensor networks*, Santa Barbara, 2006.

[8] Abidalrahman Moh'd, Nauman Aslam, William Robertson, William Phillips, "C-Sec: Energy efficient link layer encryption protocol for Wireless Sensor Networks," in *Consumer Communication and Networking Conference (IEEE-CCNC)*, Las Vegas, January, 2012.

[9] FIPS PUB 197, "Advanced Encryption Standard (AES)," National Institute of Standards and Technology, U.S. Department of Commerce, 2001.

[10] Chris Karlof, Naveen Sastry, David Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004*, Baltimore, MD, USA, November, 2004.

[11] Qi Xue, Aura Ganz, "Runtime security composition for sensor networks (SecureSense)," in *IEEE Vehicular Technology Conference, VTC*, 2003.

[12] Devesh Jinwala, Dhiren Patel, Kankar Dasgupta, "FlexiSec: A Configurable Link Layer Security Architecture for Wireless Sensor Networks," *Journal of Information Assurance and Security,* vol. 4, pp. 582-603, 2009.

[13] Castalia Simulator. Available at: http://castalia.npc.nicta.com.au/.

[14] Holger Karl, Andreas Willig , Protocols and Architectures for Wireless Sensor Networks, New York: John Wiley & Sons, 2007.

[15] Abidalrahman Moh'd, Nauman Aslam, William Phillips, William Robertson, Hosein Marzi, "SN-SEC: A Secure Wireless Sensor Platform with Hardware Cryptographic Primitives," *Journal of Personal and Ubiquitous Computing, DOI 10.1007/s00779-012-0563-9,* 2012.

[16] Zylin Processor Unit (ZPU), http://repo.or.cz/w/ zpu.git?a=blob_plain;f=zpu/docs/zpu_arch.html. Retrieved November 15, 2011.

[17] Walters, John Paul Liang, Zhengqiang Shi, Weisong Chaudhary, Vipin, Security in Distributed, Grid, and Pervasive Computing, Yang Xiao EdsCRC Press, 2007.

[18] Anthony Wood,John Stankovic, "Denial of service in sensor networks," *IEEE Computer Magazine,* vol. 35, no. 10, pp. 54-62, 2002.

[19] Wen Hu, Peter Corke, Wen Chan Shih, Leslie Overs, "secFleck: A Public Key Technology Platform for Wireless Sensor Networks," in *European Conference on Wireless Sensor Networks*, Cork, Ireland, 2009.

[20] Jorge Portilla, Andrés Otero, Eduardo de la Torre, Teresa Riesgo, Oliver Stecklina, Steffen Peter, Peter Langendörfer, "Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC Hardware Coprocessors," *International Journal of Distributed Sensor Networks,* vol. 2011, 2010.

[21] ZigBee Alliance, "ZigBee Specification," in *Technical Report Document 053474r17*, 2008.

[22] Sergei Skorobogatov, "Semi-invasive attacks - a new approach to hardware security," Technical report, University of Cambridge, Computer Laboratory, 2005.

[23] Mika Stahlberg, "Radio Jamming attacks against two popular mobile networks," Helsinki University of Tech. Seminar on Network Security, 2000.

[24] Chris Karlof, David Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Secure routing in wireless sensor networks: Attacks and countermeasures,* vol. 1, no. 2-3, pp. 293-315, 2003.

[25] Yih-Chun Hu, Adrian Perrig, David B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks," in *INFOCOM*, 2003.

[26] Lingxuan Hu, David Evans, "Using directional antennas to prevent wormhole attacks," in *NDSS*, 2004.

[27] Weichao Wang, Bharat K. Bhargava, "Visualization of wormholes in sensor networks," in *Workshop on Wireless Security*, 2004..

[28] Srdjan Capkun, Levente Buttyán, Jean-Pierre Hubaux, "Sector: secure tracking of node encounters in multi-hop wireless networks," in *SASN*, 2003.

[29] James Newsome, Elaine Shi, Dawn Song, "The Sybil Attack in Sensor Networks: Analysis and Defenses," in *Third International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004.

[30] Anthony Wood,John Stankovic, "Denial of service in sensor networks," *computer,* vol. 35, no. 10, pp. 54-62, 2002.

[31] Tuomas Aura, Pekka Nikander, Jussipekka Leiwo, "DOS-resistant authentication with client puzzles," Lecture Notes in Computer Science, 2001.

[32] David Raymond, Scott Midkiff, "Denial-of-service in wireless sensor networks: Attacks and defenses," in *IEEE Pervasive Computing*, 2008.

[33] Jing Deng, Richard Han, Shivakant Mishra, "Defending against path-based dos attacks in wireless sensor networks," in *3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '05)*, 2005.

[34] Ioannis Krontiris, Tassos Dimitriou, "Authenticated In-Network Programming for Wireless Sensor Networks," in *In Proceedings of the 5th International Conference on AD-HOC Networks and Wireless (ADHOC-NOW)*, 2006.

[35] Adi Mallikarjuna Reddy V AVU Phani Kumar, D Janakiram, G Ashok Kumar, "Operating Systems for Wireless Sensor Networks: A Survey," Distributed and Object Systems Lab, Indian Institute of Technology, Madras, India, 2007.

[36] Jason Hill, Robert Szewczyk, Alec Woo, Philip Levis, Sam Madden, Cameron Whitehouse,Joseph Polastre, David Gay, Cory Sharp, Matt Welsh,Eric Brewer, David Culler, "TinyOS: An operating system for sensor networks," In W. Weber, J. M. Rabaey, and E. Aarts, editors, Ambient Intelligence, Springer, 2005.

[37] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, David Culler, "The nesC language: A holistic approach to networked embedded systems," in *Programming Language Design and Implementation (PLDI)*, June, 2003.

[38] Raffael Bloch, "Enhanced Task Scheduling in TinyOS 2.0 Channel Allocation in AMUHR," in *Semester Thesis, Distributed Computing Group*, Zurich, Switzerland, 2006.

[39] Ian FuatAkyildiz, Mehmet Can Vuran, Wireless sensor networks, Chichester, West Sussex, U.K.; Hoboken, NJ: Wiley, 2010.

[40] Pedro Wightman, Miguel Labrador, Topology control in wireless sensor networks: with a companion simulation tool for teaching and research, New York: Springer, 2009.

[41] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, 2000.

[42] Y. Sankarasubramaniam, I. F. Akyildiz, S. W. McLaughlin, "Energy Efficiency based Packet Size Optimization in Wireless Sensor Networks," in *The First IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, Alaska, USA, 2003.

[43] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST Special Publication 800-38C," 2004.

[44] United Stated National Security Agency, "Skipjack and KEA algorithm specifications, Version 2.0," 1998.

[45] Arjen K. Lenstra and Eric R. Verheul, "Selecting cryptographic key sizes," *Journal of Cryptology,* vol. 14, no. 4, p. 255–293, 2001.

[46] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar, "SPINS: Security protocols for sensor networks," in *Seventh Annual ACM International Conference on Mobile Computing and Networks (MobiCom 2001),*, Rome, Italy, July, 2001.

[47] Ronald Rivest, "The RC5 encryption algorithm," in *1st Workshop on Fast Software Encryption,*, 1995.

[48] Mark Luk, Ghita Mezzour, Adrian Perrig, Virgil Gligor, "MiniSec: A secure sensor network communication architecture," in *In the Proceedings of ACM and IEEE Conference on Information Processing in Sensor Networks (IPSN)*, Cambridge, Massachusetts, USA, April, 2007.

[49] Phillip Rogaway, Mihir Bellare, John Black, Ted Krovetz, "OCB:A Block-Cipher Mode of Operation for Efficient Authenticated Encryption," in *ACM TISSEC*, November, 2001.

[50] Burton Bloom, "Space/time trade-offs in hash coding with allowable errors," in *In Communications of the ACM*, July 1970.

[51] Leonard Lighfoot, Jian Ren and Tongtong Li , "An Energy Efficient Link-Layer Security Protocol for Wireless Sensor Networks," in *IEEE EIT*, Chicago, USA, 2007.

[52] Shiqun Li, Tieyan Li, Xinkai Wang, Jianying Zhou, Kefei Chen, "Efficient link layer security scheme for wireless sensor networks," *Journal of Information And Computational Science,* vol. 4, no. 2, p. 553–567, 2007.

[53] DJ Wheeler, RJ Needham, "Correction of xtea," in *unpublished manuscript, Computer Laboratory, Cambridge University*, 1998.

[54] Morris Dworkin , "NIST Special Publication 800-38D. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM and GMAC)," 2007.

[55] Necla Bandirmali, Ismail Erturk, "WSNSec: A scalable data link layer security protocol for WSNs," *Ad Hoc Networks,* vol. 10, no. 1, p. 37 – 45, 2012.

[56] Mihir Bellare, Joe Kilian, Phillip. Rogaway, "The security of Cipher Block Chaining," *Advances in Cryptology, Crypto '94 Proceedings, Lecture Notes in Computer Science,* vol. 839, pp. 340-358, 1994.

[57] National Institute on Standards and Technology Computer Security Resource Center, NIST's Policy on Hash Functions http://csrc.nist.gov/groups/ST/hash/policy.html. Accessed: August, 2012..

[58] Lawrence Bassham, William Burr,Morris Dworkin, James Foti, Edward Roback, "Report on the Development of the Advanced Encryption Standard (AES)," National Institute of Standards and Technology, 2000.

[59] Murat Çakıro lu, Cüneyt Bayilmi, Ahmet Turan Özcerit, Özdemir Çetin, "Performance evaluation of scalable encryption algorithm for wireless sensor networks," *Scientific Research and Essays,* vol. 5, no. 9, pp. 856-861, 2010.

[60] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Fergusonk,Tadayoshi Kohno, Mike Stay, "The Twofish Team's Final Comments on AES selection," NIST, 2000.

[61] James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr,Morris Dworkin, James Foti, Edward Roback, "Report on the Development of the Advanced Encryption Standard (AES)," Computer Security Division, Information Technology Laboratory, NIST, 2000.

[62] Xinmiao Zhang, Keshab K. Parhi, "An FPGA-Based Performance Evaluation of the AES Algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 12, no. 9, pp. 545- 557 , 2004.

[63] Lian Huai, Xuecheng Zou, Zhenglin Liu, Yu Han, "An Energy-Efficient AES-CCM Implementation for IEEE 802.15.4 Wireless Sensor Networks," in *International Conference on Networks Security, Wireless Communications and Trusted Computing*, Wuhan, Hubei, April, 2009.

[64] Martin Feldhofer, Johannes Wolkerstorfer, Vincent Rijmen, "AES Implementation on a Grain of Sand," in *In IEEE Proc. on Information Security*, 2005.

[65] Panu Hämäläinen, Timo Alho, Marko Hännikäinen,Timo D. Hämäläinen, Design and implementation of low-area and low-power AES encryption hardware core.In Proc.9th Euro micro Conf. Digital System Design (DSD2006),Cavtat,Croatia, pp.577–583, 2006.

[66] Nele Mentens, Lejla Batina, Bart Preneel, Ingrid Verbauwhede, "A Systematic Evaluation of Compact Hardware Implementations for the Rijndael S-Box," in *in*

Proc. CT-RSA, 2005.

[67] Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger, "MD5 considered harmful today: Creating a rogue CA certificate," in *In 25thChaos Communication Congress (25C3)*, December, 2008.

[68] Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu, "Finding Collisions in the Full SHA-1," in *Advances in Cryptology, proceedings of CRYPTO 2005, Lecture Notes in Compute Science 3621*, 2005.

[69] Makoto Sugita, Mitsuru Kawazoe, Hideki Imai, "Basis Based Cryptanalysis of SHA-1," Cryptology ePrint Archive, Report 2006/098, 2006.

[70] Shai Halevi, Hugo Krawczyk, "Strengthening Digital Signatures via Randomized Hashing," *Advances in Cryptology - CRYPTO '06,* vol. 4117, pp. 41-59, 2006.

[71] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, "The Keccak SHA-3 submission," http://keccak.noekeon.org/Keccak-submission-3.pdf, 2011.

[72] "http://www.xilinx.com/products/spartan6/index.htm".

[73] Ronald Rivest, Adi Shamir, Leonard Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM,* vol. 21, no. 2, p. 120–126, 1978.

[74] Michael Rabin, "Digital Signatures and Public Key Functions as Intractable as Factoring," Technical Memo TM-212, Lab. for Computer Science, MIT, 1979.

[75] Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, "NTRU: A Ring Based Public Key Cryptosystem," in *Algorithmic Number Theory (ANTS III)*, Portland, OR, June, 1998.

[76] Ian Blake, Gadiel. Seroussi, Nigel Smart, Elliptic Curves in Cryptography, Cambridge University Press, Cambridge University Press, 2000.

[77] National Institute of Standards and Technology. Recommended Elliptic Curves for Federal Government Use. August, 1999..

[78] Holger Karl, Andreas Willig, Protocols and Architectures for Wireless Sensor Networks, John Wiley & Sons: Chichester ISBN: 978-0-470-09510-2, June, 2005, p. 44.

[79] Peter Montgomery, "Speeding the Pollard and Elliptic Curve Methods of Factorizations," *Math. Comp.,* vol. 48, pp. 243-264, 1987.

[80] Daniel Bernstein, Tanja Lange, "Faster addition and doubling on elliptic curves," *Advances in Cryptology ASIACRYPT 2007, Lecture Notes in Computer Science 4833,* 2007.

[81] George Robert Blakley, "A Computer Algorithm for Calculating the Product A.B

modulo M," *IEEE Transactions on Computers,* Vols. C-32, no. 5, p. 497–500, May 1983.

[82] Paul Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," *Advances inCryptology - Crypto'86, LNCS,* vol. 263, p. 311–323, 1987.

[83] Peter Montgomery, "Multiplication without trial division," *Mathematics of Computation,* vol. 44, p. 519–521, 1985.

[84] David Narh Amanor, Christof Paar, Jan Pelzl, Viktor Bunimov, Manfred Schimmler, "Efficient hardware architectures for modular multiplication on FPGAs," in *Proc. 15th Int'l Conf. Field Programmable Logic and Applications (FPL '05)*, 2005.

[85] Miroslav Knezevi , Lejla Batina, Ingrid Verbauwhede, "Modular Reduction without Precomputational Phase," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, May, 2009.

[86] Standards for Efficient Cryptography Group (SECG). Recommended Elliptic Curve Domain Parameters. SEC 2, september, 2000.

[87] Certicom ECC Challenge. November 10, 2009 update. http://www.certicom.com/images/pdfs/challenge-2009.pdf.

[88] Fact Sheet NSA Suite B Cryptography, U.S. National Security Agency, (2009). http://www.nsa.gov/ia/programs/ suiteb_cryptography/index.shtml .

[89] Daniel Bernstein, Irrelevant patents on elliptic-curve cryptography. http://cr.yp.to/ecdh/patents.html. Retrived August 2011.

[90] Certicom, "http://www.ecc-challenge.info/," retrived Feb, 15, 2013.

[91] Certicom research, "SEC 2: Recommended Elliptic Curve Domain Parameters," Standards for Efficient Cryptography Group (SECG), 2010.

[92] Lejla Batina, Jorge Guajardo, Bart Preneel, Pim Tuyls, Ingrid Verbauwhede, "Public-Key Cryptography for RFID Tags and Applications," in *RFID Security: Techniques,Protocols and System-On-Chip Design*, 2008.

[93] Erdinc Ozturk,Berk Sunar, E. Savas, "Low-Power Elliptic Curve Cryptography Using Scaled Modular Arithmetic," in *Cryptographic Hardware and Embedded Systems - CHES*, 2004.

[94] Gunnar Gaubatz, Jens-Peter Kaps, Erdinc Ozturk, Berk Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," in *Third IEEE International Conference on Pervasive Computing and Communications Workshops, Workshop on Pervasive Computing and Communications Security–PerSec'05*, 2005.

[95] Sandeep S. Kumar, Christof Paar, "Are standards compliant Elliptic Curve

Cryptosystems feasible on RFID?," in *Workshop on RFID Security*, Graz, Austria, 2006.

[96] Harald Aigner,Holger Bock, Markus Hutter, Johannes Wolkerstorfer, "A low-cost ECC coprocessor for smartcards," in *Cryptographic Hardware and Embedded Systems - CHES 2004, LNCS 3156*, 2004.

[97] Yong Ki Lee, Kazuo Sakiyama,Lejla Batina, Ingrid Verbauwhede, "Elliptic Curve Based Security Processor for RFID," *IEEE Transactions on Computer,* vol. 57, no. 11, p. 1514–1527, 2008.

[98] Pim Tuyls, Lejla Batina, "RFID-tags for Anti-Counterfeiting," *Topics in Cryptology - CT-RSA,* vol. 3860, pp. 115-131, 2006.

[99] Lejla Batina, Nele Mentens, Kazuo Sakiyama,Bart Preneel, Ingrid Verbauwhede, "Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks," in *Third European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, Hamburg, Germany, 2006.

[100] Junqi Zhang, Vijay Varadharajan, "Wireless sensor network key management survey and taxonomy," *Journal of Network and Computer Applications,* vol. 2010, no. 33, p. 63–75, 2010.

[101] Siquan Hu, Mehul Motani, "Early Overhearing Avoidance in Wireless Sensor Networks," *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet,* vol. 4982, pp. 26-35, 2008.

[102] Marcos A. Simplicio Jr, Bruno T. de Oliveira, Cintia B. Margi, Paulo S.L.M. Barreto, Tereza C.M.B. Carvalho, Mats Näslund, "Survey and comparison of message authentication solutions on wireless sensor networks," *Ad Hoc Networks,* 2012.

[103] Paolo Santi, Janos Simon, "Silence is Golden with High Probability:Maintaining a Connected Backbone inWireless Sensor Networks," *Lecture Notes in Computer Science,* vol. 2920, pp. 106-121, 2004.

[104] James Nechvatal,Elaine Barker, Lawrence Bassham, William Burr,Morris Dworkin, James Foti, Edward Roback, "Report on the Development of the Advanced Encryption Standard (AES)," National Institute of Standards and Technology, October 2, 2000.

[105] Abidalrahman Mohammad, H. Marzi, N. Aslam and L. Tawalbeh, "Hardware Implementation of Secure hasing Functions on FPGAs for WSNs," in *Third International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, Istanbul, Turkey, July, 2010.

[106] Sergei Skorobogatov, "Physical Attacks on Tamper Resistance: Progress and Lessons," in *2nd Army Research Office Workshop on Hardware Assurance*, Washington, DC, USA, April, 2012.

[107] Omnetpp. Available at: http://www.omnetpp.org/.

[108] Gunter Bolch, Stefan Greiner, Hermann de Meer, Kishor Shridharbhai Trivedi, Queueing Networks and Markov Chains : Modeling and Performance Evaluation With Computer Science Applications, Wiley-Interscience, 2006.

[109] Rkjane Forri, "The strict avalanche criterion: spectral properties of booleans functions and an extended definition," *Advances in cryptology, Crypto'88, Lecture Notes in Computer Science,* vol. 403, p. 450–468, 1990.

[110] Walid Y. Zibideh, Mustafa M. Matalgah, "Modified-DES Encryption Algorithm with Improved BER Performance in Wireless Communication," in *Proceedings of the 2011 IEEE Radio and Wireless Symposium (RWS 2011) part of the Radio Wireless Week (RWW 2011)*, Phoenix, AZ, USA, January, 2011.

[111] Mohamed A. Haleem, Chetan Nanjunda Mathur, Rajarathnam Chandramouli, K. P. Subbalakshmi, "Opportunistic Encryption: A Trade-Off between Security and Throughput in Wireless Networks," *IEEE Transactions on Dependable and Secure Computing,* vol. 4, no. 4, p. 313–324, 2007.

[112] E. Blaß, M. Zitterbart, "Efficient Implementation of ECC for Wireless Sensor Networks," Telematics Technical Reports, University of Karlsruhe, March 2005.

[113] Joseph Bonneau, Ilya Mironov, "Cache-Collision Timing Attacks Against AES," in *Cryptographic Hardware and Embedded Systems CHES*, Yokohama, Japan, October, 2006.

[114] M. Healy, T. Newe, and E. Lewis, "Efficiently securing data on a wireless sensor network," *Journal of Physics: Conference Series,* vol. 76, no. 1, 2007.

[115] Edward Sazonov, Ratneshwar Jha , Kerop Janoyan, Vidya Krishnamurthy, Michael Fuchs, Kevin Cross, "Wireless Intelligent Sensor and Actuator Network (WISAN): a scalable ultra-low-power platform for structural health monitoring," in *Health Monitoring and Smart Nondestructive Evaluation of Structural and Biological Systems*, 2006.

[116] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul,Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, Edward W. Felten, "Lest We Remember: Cold Boot Attacks on Encryption Keys," in *Proc. 17th USENIX Security Symposium*, San Jose, CA, 2008.

[117] NIST, FIBS-PUB 180-2,Secure Hash Standard, August, 2002.

[118] Katsuyuki Okeya, Hiroyuki Kurumatani, Kouichi Sakurai, "Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications," *Lecture Notes in Computer Science,* vol. 1751, pp. 238-257, 2000.

[119] Viktor Bunimov, Manfred Schimmler, "Area and Time Efficient Modular Multiplication of Large Integers," in *IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, 2003.

[120] Cetin K. Koc, Tolga Acar , "Montgomery multiplication in GF(2k)," *Designs, Codes and Cryptography.,* vol. 14, no. 1, pp. 57-69, 1998.

[121] Darrel Hankerson, Alfred Menezes, Scott Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag, 2004.

[122] Marcelo E. Kaihara, Naofumi Takagi, "A VLSI algorithm for modular multiplication/division," in *IEEE 16th Symposium on Computer Arithmetic*, Los Alamitos, California, 2003.

[123] Chang Hoon Kim, Soonhak Kwon, Jong Jin Kim, Chun Pyo Hong, "A compact and fast division architecture for a finite field GF(2m)," in *The international conference on Computational science and its applications*, 2003.

[124] Virantha N. Ekanayake, Clinton Kelly, Rajit Manohar, "BitSNAP: Dynamic Significance Compression For a Low-Energy Sensor Network Asynchronous Processor," in *Proceedings of the 11th International Symposium on Asyncronous Circuits and Systems*, March 2005.

[125] Joyce Kwong Yogesh Ramadass, Naveen Vermal, Markus Koeslerl, Korbinian Huber, Hans Moormann, Anantha Chandrakasan, "A 65nm Sub-Vt, Microcontroller with Integrated SRAM and Switched-Capacitor DC-DC Converter," in *IEEE International Solid-State Circuits Conference (ISSCC)*, February 2008.

[126] Bo Zhai, Leyla Nazhandali, Javin Olson, Anna Reeves, Michael Minuth, Ryan Helfand,Sanjay Pant, David Blaauw, Todd Austin , "A 2.60 pJ/Inst subthreshold sensor processor for optimal energy efficiency," in *IEEE Symposium on VLSI Circuits (VLSI-Symp)*, June 2006.

[127] David J. Malan, Matt Welsh, Michael D. Smith, "Implementing Public-Key Infrastructure for Sensor Networks," *ACM Transactions on Sensor Networks,* vol. 4, no. 4, 2008.

[128] Kui Ren, Kai Zeng, Wenjing Lou, Patrick J. Moran, "On Broadcast Authentication in Wireless Sensor Networks," in *IEEE Transactions on Wireless Communications*, 2007.

[129] Osman Ugus, Dirk Westhoff, Ralf Laue, Abdulhadi Shoufan, Sorin A Huss, "Optimized Implementation of Elliptic Curve Based Additive Homomorphic Encryption for Wireless Sensor Networks," in *2nd Workshop on Embedded Systems Security*, 2007.

[130] "Artix-7 FPGAs Data Sheets," Xilinx Corporation, 2013.

Appendix A –Sample run for encryption primitives

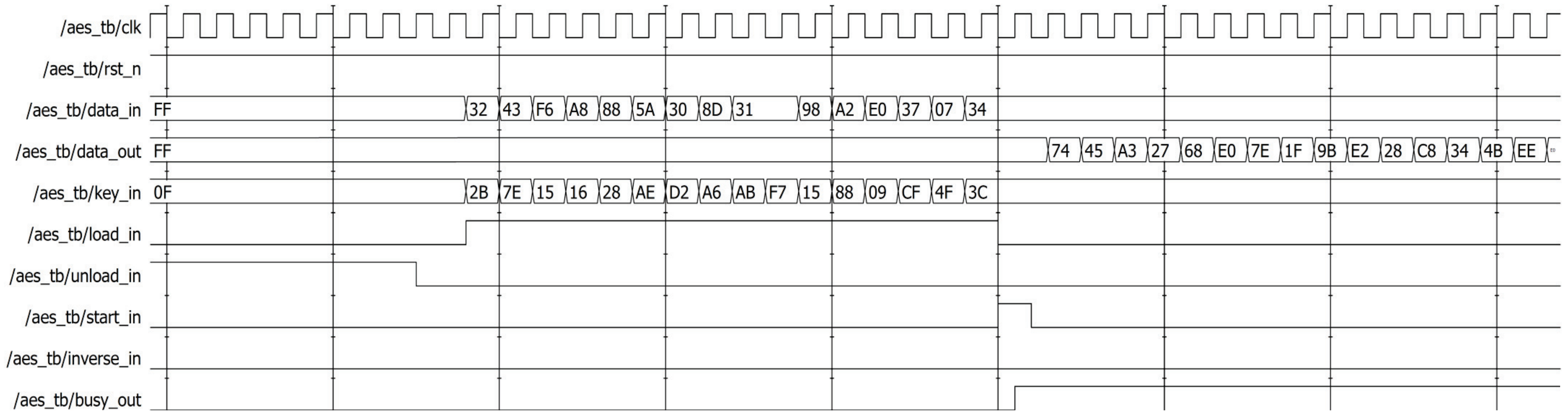


Figure A.1 AES input

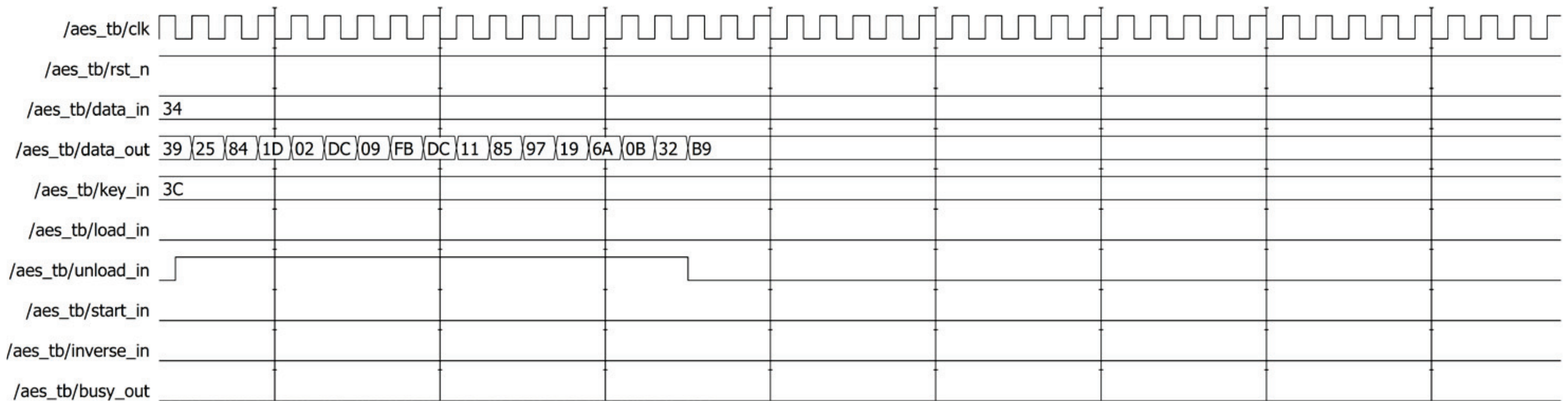


Figure A.2 AES output

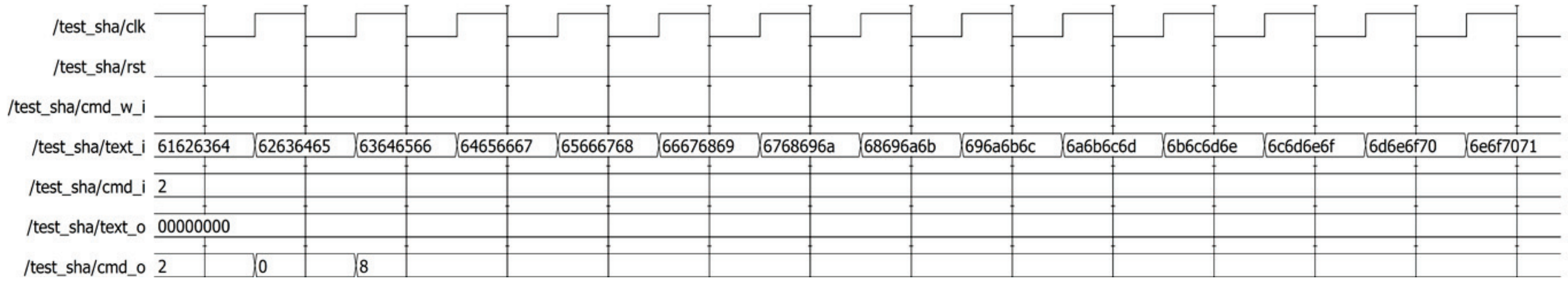


Figure A.3 SHA-256 input

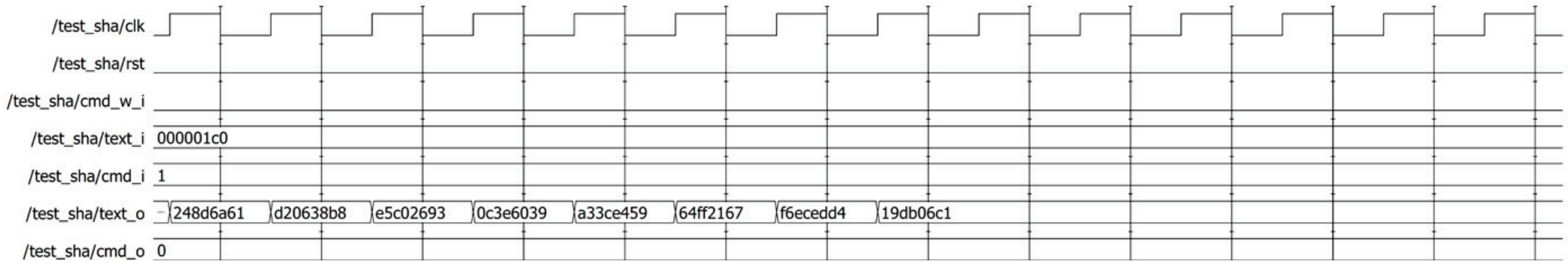


Figure A.4 SHA-256 output

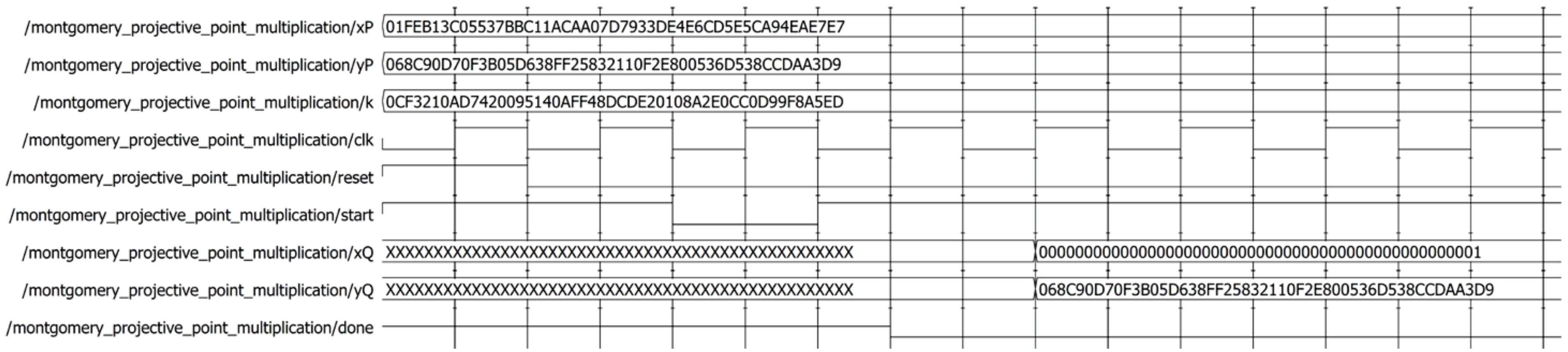


Figure A.5 Elliptic curve point multiplication input

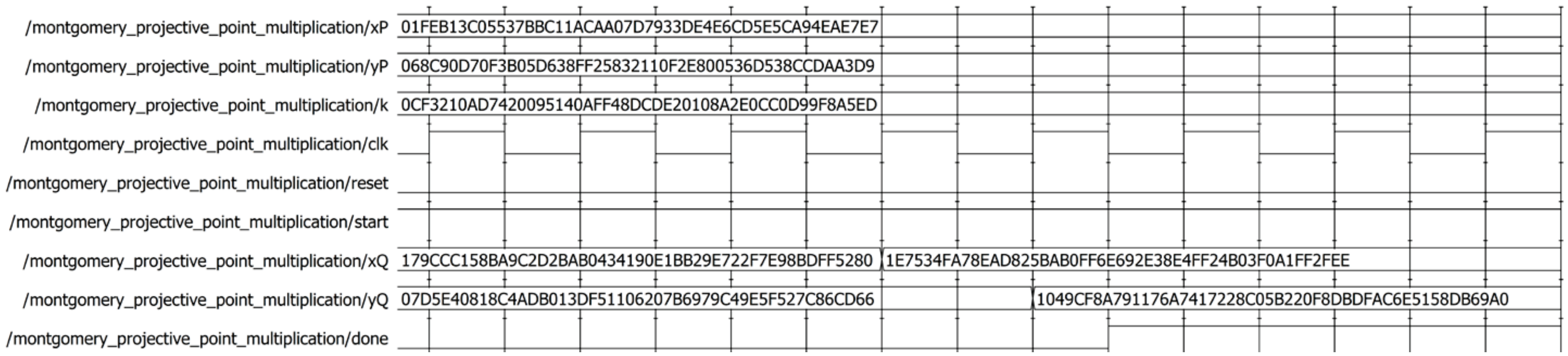


Figure A.6 Elliptic curve point multiplication output