Masters in Informatics Engineering
Dissertation
Final Report

Interoperable Security Model for Wireless Sensor Networks

André Sérgio Nobre Gomes
asng@student.dei.uc.pt

Advisor:
Jorge Miguel Sá Silva

Co-Advisor:
António Jorge da Costa Granjal

Date: July 12th, 2012

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Abstract

In today's world, Wireless Sensor Networks field of study is widely discussed among the research community, being expected that this trend will continue further. However, and even though some valid proposals already exist, security mechanisms to apply to these networks are still in a very embryonic state. Together with this lack of stable and proven mechanisms, there is also a void when it comes to establish secure communication between different devices with distinct architectures and operating systems.

In this dissertation, the main goal is to develop a new security model that can be applied to a whole range of resource constrained devices while maintaining compatibility and interoperability with different systems using compatible IPv6 stacks. Additionaly, this work intends to provide an abstraction layer for the security mechanisms, meaning that at each communication layer can easily be secured through the use of simple functions.

Finally, this work also intends to provide a contribution that can be applied to ICT FP7 GINSENG - Performance Control in Wireless Sensor Networks in order to provide security to its topology without compromising the strict timing and resources utilization requirements.

Keywords

"Contiki Security", "Criptography", "GINSENG", "Interoperable Security Model", "Key Distribution", "Security in Wireless Sensor Networks", "Security Layer for Wireless Sensor Networks","Security Model for Wireless Sensor Networks", "TinyOS Security".

Index

List of Figures

List of Tables

Acronyms

3DES	Triple Data Encryption Standard
AAA	Authentication, Authorization, and Attack Detection
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
API	Application Programming Interface
BLIP	Berkeley Low-power IP
CBC	Cipher-Block Chaining
CCM	Counter with CBC-MAC
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Access
CTR	Counter
DH	Diffie-Hellman
DoS	Denial of Service
DAS	Digital Signature Algorithm
EC-DH	Elliptic Curve Diffie-Hellman
FFD	Full-Function Device
FIFO	First In First Out
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6	Internet Protocol version 6
IV	Initial Vector
KDC	Key Distribution Center
LED	Light-Emitting Diode
MAC (Address or TCP/IP Layer)	Media Access Control
MAC (Authentication)	Message Authentication Code
MCU	Micro Controller Unit
MD5	Message-Digest algorithm 5
MIC	Message Integrity Code
MIPv6	Mobile Internet Protocol version 6
ND	Neighbor Discovery
OS	Operating System
PAN	Personal Area Network
QoS	Quality of Service
RAM	Random Access Memory
RF	Radio Frequency
RFC	Request For Comments
RFD	Reduced-Function Device
ROM	Read Only Memory

RSA	Ron Rivest, Adi Shamir and Leonard Adleman
RX	Receiver
SHA-1	Secure Hash Algorithm 1
SHA-2	Secure Hash Algorithm 2
SICS	Swedish Institute of Computer Science
SKKE	Symmetric-Key Key Exchange
SNMP	Simple Network Management Protocol
SYN	Synchronize Packet
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol and Internet Protocol
TDMA	Time Division Multiple Access
TTL	Time To Live
TX	Transmitter
UDP	User Datagram Protocol
USA	United States of America
USB	Universal Serial Bus
WSN	Wireless Sensor Network
XOR	Exclusive OR

Chapter 1
Introduction

The research community is not unanimous regarding how Wireless Sensor Networks (WSNs) first started. Although most researchers point out that the first real deployment was set in 2002 in Great Duck Island, Maine, USA, some such as Matt Welsh from the University of Harvard, USA claim that it was the first modern WSN but not the first, with the first being a project from the United States Air Force called Igloo White that was deployed in 1967 in Vietnam [1].

Igloo White was a project with a single goal – track the activity and vehicle movement in Ho Chi Minh Trail, Laos, a vital supply route to South Vietnam. In order to provide such monitoring, military aircrafts deployed look-alike missile devices with disguised antennas and seismic/acoustic sensors. To collect data, another aircraft would follow the trail and download data from each sensor, allowing the military to detect any activity around the sensors. Even though it was in 1967, these devices had enough power from lithium batteries to run continuously for 30 days, which today is still a pretty good autonomy considering that power consumption and battery runtimes are modern times issues in WSNs.

However, Igloo White was not a big boost to the concept and technology of WSNs. This is a symptom of times, as in 1967 computing was not evolved enough to support this kind of devices as today. Therefore, there is some credit to the researchers that claim that Great Duck Island was really the first deployment of modern days WSNs. In fact, this deployment to simply monitor environmental conditions in order to study nesting behavior of seabirds was the processor of the WSNs boom. From this deployment, concepts such as mote (sensor node) obtained a new meaning, with thousands of researchers across the globe interested in the whole research area that was created with WSNs. Moreover, and even more important, small operating systems (OS) fully designed to WNSs which were still in a very embryonic state such as TinyOS [2] (first released in 2000) and more recently Contiki [3] (first released in 2008) earned the importance and relevance that today makes them the most used OSs for WSNs with thousands of contributors to improve their cores and modules due to the open-source concept used in desktop OSs such as Linux.

More recently, with an enormous number of researchers working in WSNs providing meaningful contributions and with the development of technology in general, new deployment scenarios became possible. These range from oil refineries to health, from cities monitoring to environmental studies in remote locations and so on. However, some of them are so critical due to their goals and safety risks that new requirements for WSNs emerge and need to be taken in account. One of these requirements, and most of the times left apart almost until the final deployment, is security. With today's wars becoming each time more virtual and with security being a major concern in our world, critical operation scenarios demand not only privacy but also a complete security system to maintain key factors such as confidentiality of data, confidence in gathered sensor values/actuation orders and reliability assurance. But there is a key issue in these requirements: there are not solutions to completely and effectively deal with security concerns.

Security in WSNs, although it is a topic which already has a lot of proposals and studies, is still in a very embryonic state without meaningful solutions that can be widely adopted and standardized. This last observation has one major cause – security in this kind of networks is not as simple and easy to implement as in traditional networks. This can be explained by

some factors that are inherent to WSNs. For example, motes are typically powered by AA batteries, which under high resources utilization tend to have a very low autonomy. And high resources utilization can be a symptom of security itself caused by another inherent factor of WSNs: cryptography and related algorithms are designed to modern computers in order to provide a high level of security, meaning that the resources utilization is also high and not adapted to constrained devices such as WSNs motes (8MHz Central Processing Unit (CPU), 10KB of Random Access Memory (RAM), and others as explained in next chapter). Also, as referred before, most of the scenarios with high security requirements are not typical scenarios but critical scenarios where factors such as real-time operation, integrity, long battery autonomy and reliability are mandatory.

Another important gap left by current solutions for WSNs security concerns compatibility and interoperability. Most proposals from the research community tend to focus in a specific platform with a specific OS, which can be explained by the difficulty of finding solutions that can be widely applied to several architectures and to different OSs designs. This difficulty is not only from the technical point of view of security, but also a consequence of the lack of standardization and normalization in the communication stacks for WSNs, which nowadays is being tackled by the Internet Engineering Task Force (IETF) with efforts to deploy an adapted Internet Protocol (IP) stack for WSNs.

Bearing these last paragraphs in mind, and considering that a big influence to this dissertation is an European project (on which the author of this dissertation has participated) that intends to apply WSNs in critical scenarios, the need for a security model that can both be applied to WSNs without compromising their normal operation and that is compatible and interoperable between different architectures and OSs emerges as a necessity.

Therefore, this dissertation aims to develop a security model that can be applied to a whole range of different architectures/equipments without compromising their normal operation and assuring that communication is done in a secure way. Moreover, it is intended that the developed security model is portable for multiple architectures, lightweight, easy to use, transparent for the common programmer and modular so that it can be configurable and able to adapt to any possible scenario of use.

Finally, this report is structured as follows. In chapter 2, a comprehensive overview regarding the state of the art of WSNs and the current proposals to address security is presented. In chapter 3, the proposal itself is described and its objectives and possible applications are also explained. In chapter 4, the validation tests for the proposed security model are defined and described with results to support them. Then, chapter 5 gives an overview regarding the work plan during the 2 semesters and the methodology that has been used. Finally, chapter 6 concludes this dissertation and presents challenges and problems which are still to overcome in future work.

Chapter 2
State of the Art

2.1. Wireless Sensor Networks – Overview

WSNs are one of the most important technologies in 21st century and usually pointed as the missing extension to connect the virtual and the real worlds [4]. Constituted by low power and low cost small nodes, WSNs have been projected for thousands of applications in several areas. Most of the researchers and technological analysts believe that in the near future micro sensors will be everywhere: our homes, factories, bodies, animals, cars or rivers. However, there are several challenges and problems that must be overcome before WSNs achieve the reliability to become a reference in industry.

Each sensor node on a WSN is usually constituted by a set of small components such as RF transceivers, indicator LEDS, low power microcontrollers with extended memory, optional sensors attached, one antenna, flash memory, some extension connectors/jumpers, one USB connector and typically a battery pack attached below the main circuit board as shown in figure 1.

Figure 1 - Example of a sensor node [5]

Self-configuration and self-organization capabilities of the sensor nodes are important requirements of WSNs, which allow the deployment of networks in an easy way without individual or manual configuration of each mote. The low cost of each sensor node is an important advantage for this type of networks, allowing economical large-scale deployment of nodes. However, the low cost together with the small dimensions nodes, causes some limitations at levels like processor capability, storage capacity, communication and mainly at power level.

On WSNs it is commonly used a central node with larger capability, usually called the Sink Node. The Sink Node is responsible for getting data cached by each node, sending this data to the remote center. Hence, WSNs require an efficient protocol to support the communication between nodes and the Sink Node and/or between neighbor nodes. Thus, WSNs can work over two different topologies, star or mesh, allowing efficient energy rationalization in a dynamic way.

To provide a good communication scheme, these network devices may use a specific protocol developed for devices with limited capabilities, the IEEE802.15.4 [6] - a Physical and MAC Layer protocol included in the ZigBee Alliance [7], which contributes for the

optimization of wireless networks for Personal Areas Networks (PAN) with low resources, guaranteeing the interoperability between different constructors and the standard.

IEEE802.15.4 is a set of standards designed for low complexity, low power consumption, short communication range and low data rates over low cost devices. This standard has a maximum physical layer (layer 1) packet of 127bytes and MAC Layer (layer2) of 102 octets. Additionally, MAC layer supports security mechanisms, offering up to AES CCM-128 based encryption/authentication, imposing an overhead of 21 octets, leaving only 81 octets for data packets. IEEE802.15.4 supports also two MAC addresses: 16-bit short and IEEE 64-bit extended. Regarding bandwidth, IEEE802.15.4 starts with 20 kbps at 868MHz, 40Kbps at 915MHz, and reaching 250Kbps at 2.4GHz.

In order to provide the support for IPv6 to implement a seamless and global connectivity, IETF created a layer 3 protocol called 6LowPAN [9]. 6LowPAN works over the IEEE802.15.4 and aims to make possible the use of upper layer protocols from conventional networks in LowPANS (UDP, TCP, HTTP, etc.).

Thus, in order to support a reliable Personal Area Network, there are different entities over IEEE802.15.4. All networks have one unique PAN coordinator, a node responsible for all networks, acting as the interface with the exterior. As mentioned before this entity is the Sink Node. Other two different devices constitute the PAN: the reduced-function devices (RFD) and the full-function devices (FFD). FFDs are devices with more powerful capabilities than RFDs, having the ability to operate as routers, indispensable function in mesh topologies. RFDs are only end nodes, with limited capabilities. In WSNs FFDs can work as PAN coordinators (Sink Node), coordinators or as end-nodes. RFDs, in turn, can just work as end-nodes. These three different devices allow the constitution of the two mentioned topologies, star and mesh. Figures 2 and 3 present these topologies and respective devices.

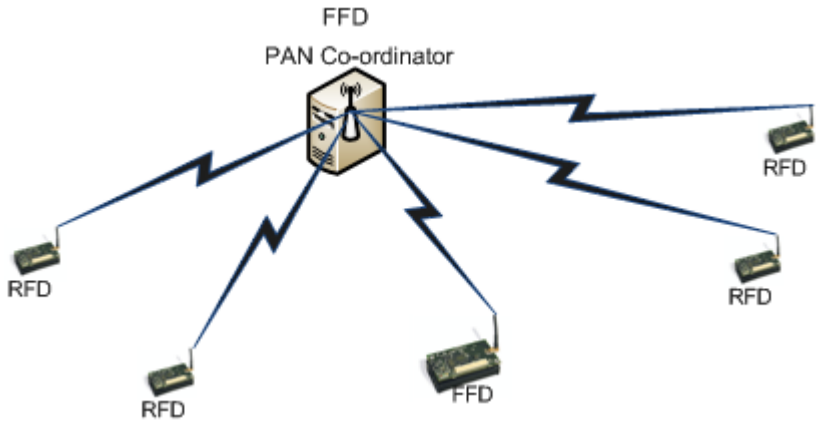

Figure 2 - Star Topology

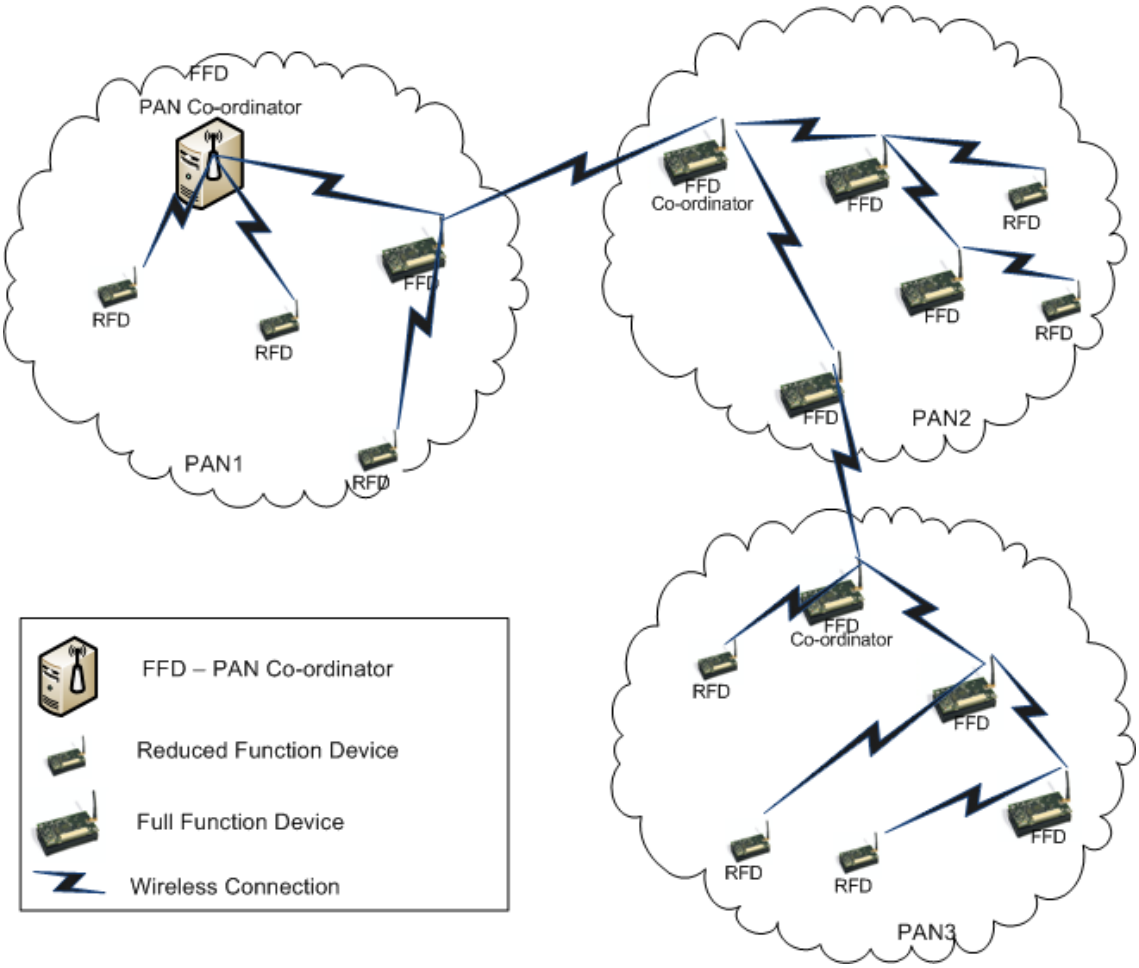

Figure 3 - Mesh Topology

As it is presented in figure 2, the star topology is the simplest, connecting all nodes directly to the PAN coordinator, as a central unit. This topology is easy to configure, deploy and manage. However, it only works fine for simple scenarios, where all devices are physically located near the PAN coordinator.

In complex scenarios, nodes can be further away from PAN coordinator than in figure 2. Figure 3 approaches this issue, showing how the nodes are connected. As it is possible to observe, there are three sub-PANs and only one PAN coordinator. It is necessary to deploy a suite of FFDs across the entire network to ensure a reliable and efficient network. For each area there is one FFD working as coordinator, which is the boundary node of that specific area. Figure 3 represents a mesh topology organized in clusters, where each node is connected to the near neighbor, spending less energy and increasing the monitored area, when comparing with solutions based on direct connection to the PAN coordinator.

To sum up, WSNs are networks constituted by simple and limited devices, with capability for measuring specific values and processing and transmitting them to a central unit. WSN devices usually rely on IEEE802.15.4 standard to implement the communication at physical and MAC Layer, allowing two different topologies, star and mesh.

Besides lifetime, one of the present problems is the lack of reliability and performance control of WSNs. It is believed that as soon as they provide performance-controlled

capabilities, they will be used massively, replacing conventional and wired technologies. Under specific characteristics like resources constrained nodes, data redundancy, limited communication protocols, complex duty cycle schemes, high number of network elements or highly dynamic scenarios, quality of service (QoS) is not only complex but also mandatory. Chen et al. [9] and Younis et al. [10] identified the QoS requirements of WSNs. Through the last years researchers have been working mainly on duty cycle schemes, energy-aware routing algorithms and communication protocols. Since most of the WSNs applications are intended to critical scenarios with real-time demanding, as for indication as for control, QoS is highly mandatory.

In fact, as WSNs will be used in a wide range of scenarios where reliability and privacy of transmitted data are critical factors, security is also a topic that needs to be taken in account and explored to fit the needs of this type of networks. Nowadays, mainly due to the limited resources of WSNs, security it is not implemented or, if so, it is implemented in a very basic way that makes WSNs vulnerable to many of the usual attacks. As WSNs may be deployed in environments like hospitals or oil refineries, there is the need to prevent and even block a node from sending false data into the system, so only authorized nodes may join or connect to the network. Another important requirement is the support of mobility since most of the current Internet devices present some type of mobility and it is expected that, if in fact sensor nodes will be everywhere, most of them also require mobility. For instance, in healthcare patients' monitoring mobility is highly desired. Heart rate, breath rate, blood pressure or weight are common factors that must be constantly monitored. Therefore, using portable systems based on WSNs make possible not only more efficient medical service, but also new freedom for the patients. Senior citizens can also benefit from mobility of WSNs. Usually eldercare systems are provided in special places, forcing people to move from their own homes. With WSNs supporting mobility, it will be possible to keep senior citizens at home for more time, guaranteeing in real time their health good condition. Mobility in WSNs will also be a key factor in oil refineries, chemistry and petro-chemistry industries among others as it will allow employees working within hazardous areas, to be safely monitored. In such places employees can be under difficult and unhealthy conditions and therefore suffer health consequences. With mobility support in WSNs, employees can do their normal work knowing that if any uncommon situation in their health state is detected, an alarm will be triggered in real time. However, this also poses challenges to security as moving nodes need to authenticate and deauthenticate often, and it is also more probable that a mobile node is replicated due to its mobile character.

Also known as one of the key factor of the "Internet of Things", WSNs, independently of all the recognized potential, remain limited in the access. The traditional WSNs protocols don't provide the flexibility and interoperability that WSNs demand. As a result, to allow the communication between conventional networks and WSNs, it is necessary to provide specific and complex gateways. However, if we integrate the IPv6 directly in sensor nodes, this will not be needed. IPv6 in sensors makes available end-to-end connectivity, without gateways or other type of translator entity between networks. Therefore, IPv6 in sensors make possible the use of end-to-end transport protocols, supporting reliability on communication and controlled performance. Besides, it will be possible to benefit from several advantages directly related to the concept of IP networks.

The use of well-known protocols like ICMP, SNMP or UDP in WSNs are an added value, as well as the possibility to use Neighbor Discovery (ND) or Mobile IPv6 (MIPv6) converging to Plug and Play networks. Integrating IPv6 in WSNs offers communication between any device connected to the Internet and a specific IPv6 enabled sensor.

Integrating IPv6 in WSNs also offers a transparent and easy approach for developing new generation applications. According to 4th Generation Networks there will be a wide variety of terminals and devices with different capabilities. Although IP has a lot of limitations, it is the common "language" that every device recognizes.

The use of IPv6 in sensors presents some drawbacks, mainly due to the native MTU of 1280 bytes. Considering, the maximum 102bytes of the IEEE802.15.4 payload, some adaptations must be performed to support IPv6 in WSNs. The 6lowPAN Work Group is an IETF working group that is responsible to adapt IPv6 to WSNs. Two RFCs were released: the RFC 4919 surveys the problematic of using IPv6 over IEEE802.15.4 and the RFC4944 presents the first adaptation model, introducing a new method to code and compress IPv6 header from 40 bytes to a minimum of 2 bytes. The same RFC also introduces the concept of fragmentation and assembly for packets whose length is bigger than 102bytes.

Although IP may bring many advantages for WSNs, namely the compatibility and interoperability of communication stacks, it also poses some concerns. In this particular case, security flaws inherent to IP (SYN flooding, IP Spoofing, Connection Hijacking, etc.) are also brought to WSNs when using IP. Thus, security mechanisms have to be implemented to deal with these concerns.

Regarding multi-hop, RFC4944 also defines that all the mesh management process must be done under 6lowPAN and, in turn, the entire route activity must be processed over 6lowPAN. This draft defines link local and global addresses, through the EUI 64bits MAC Address from IEEE802.15.4.

2.2. Wireless Sensor Networks – Operating Systems

Nowadays, operating systems play a main role in WSNs. In fact, and even though some particular scenarios do not use OSs to save resources, it is becoming rare to find deployments which do not rely on an OS as the basis for the application itself.

The use of OSs can be evaluated as typically beneficial, mostly because the interface between the application and the whole underlying architecture is already provided with easy to use Application Programming Interfaces (APIs) and the management of the resources is not another task for the programmer to deal with. But other reasons can be considered as advantages of their use. For instance, implement communication stacks and standards enable some interoperability and compatibility between different systems/architectures and even between different OSs. However, an improvement never comes without drawbacks. In this case, OSs can be considered as overhead, meaning that resources that already are limited will be more limited for the application itself because the OS has its needs to provide its functionalities. Moreover, some may also consider that OSs may limit the programmer to high level interfaces and that the diversity of available solutions cause compatibility issues as protocols and functions tend to be implemented in different ways depending of the developer.

For this dissertation, two different and meaningful OSs are taken in account: Contiki and TinyOS. They have been chosen because they are the most widely adopted in Europe and United States of American, respectively.

2.2.1. Contiki

Contiki [11] is an open source, highly portable and multi-tasking OS initially developed by Adam Dunkels at the Swedish Institute of Computer Science in 2004 and first released in 2005. This OS was designed for WSNs from scratch, being an event-driven OS with optional preemptive multitasking.

As advantages to already existing OSs, Contiki brought TCP/IP (including IPv6/6LowPAN)) support for WSNs, a new type of threads called proto-threads which are extremely lightweight and stackless, a kernel prepared to a wide range of architectures and dynamically linked code.

It is also important to refer that Contiki can run on very constrained devices, with a typical Contiki configuration averaging 2 kilobytes of RAM and 40 kilobytes of ROM.

Regarding the programming itself, it is plain C language with some adjustments to meet the OS structure. For instance, in each program the programmer needs to define the processes and to call the function that will start each one.

Finally, Contiki is currently in version 2.5 with support for the latest platforms and protocols already embedded in its core.

2.2.2. TinyOS

TinyOS [12] is another open source component-based OS. It first started as collaboration between the University of California, Berkeley, USA, Crossbow Technology and Intel Research in 1999, and its first version has been released in 2000 with version 1.0.

As key advantages, it proposed a whole new operating system specially designed for WSNs that features an event-driven kernel with non-preemptive multitasking and completely non-blocking. In fact, each task is run in FIFO (First In First Out) order and programs are built out of software components which already exist or are defined by the programmer.

Currently, TinyOS is the leading operating system for WSNs in United States of American and it has one major advantage over any other operating system for WSNs in the world – its enormous supporting community.

As for the programming itself, TinyOS uses statically linked code and both applications and libraries are written using a language that is very similar to C called nesC [13].

At the present date, TinyOS latest version was 2.1.1 which has been released in 2010 and counts with hundreds of new features since its first version, being the most important the support for IPv6 (6lowPAN) through a module called Berkeley Low-power IP stack (BLIP) developed almost entirely by Jonathan Hui.

2.3. Wireless Sensor Networks – Security

2.3.1. Overview

Obstacles to WSNs Security

When considering security for WSNs, one needs to take in account that there are several obstacles that are specific of this type of networks and also related with the devices used to build them [19].

The first big obstacle is that WSNs have very limited resources in terms processing, memory and power.

Regarding processing, typically the microprocessors which are embedded in sensor nodes are 16 bits architectures with not more than 8MHz of clock speed. Hence, complex security algorithms which are computing intensive cannot be used without compromising the whole system.

In terms of memory and storage space, we have to bear in mind that a sensor node is a tiny device, with a small amount of RAM and ROM (for code storage). Therefore, the security related code must be small and adapted to the available resources.

Finally, in terms of power we have to assume that when a node is deployed, it cannot be easily replaced or recharged (low maintenance is one of the goals). As it runs on batteries, the security mechanisms cannot use much energy in order to save battery life.

Another key factor in WSNs it is their unreliable communication. This can be divided into three different aspects:

- ✓ Unreliable Data Transfer: usually the packet-based routing of sensor networks is connectionless and thus inherently unreliable (UDP is used most of the times). This means that some critical security packets may get lost or damaged.

- ✓ Conflicts: because of the broadcast nature of a WSN, packets may collide in an high density sensor network where the nodes share the middle of transfer even with efficient MAC protocols with collision detection and collision avoidance. This leads to an high rate of packet loss.

- ✓ Latency: the multi-hop routing, the node processing, the network congestions and also the low bandwidth can lead to high latencies and therefore to problems in node synchronization, which is critical for some security mechanisms.

Finally, another aspect that is particular of WSNs and may cause security issues and obstacles is their unattended operation.

The first reason is that sensor nodes are usually deployed in the field within an exposed environment without protection from attackers. This leads to multiple security threats, namely a whole range of physical attacks.

The second it that one of the objectives of WSNs is to minimize deployment and maintenance costs, which leads to remote management after deployment. Considering this, there is no way of detecting physical tampering or maintenance issues. Thus, the physical attacks referred above are almost impossible to detect/avoid.

Finally, their distributed character means that there is not central management point. Although being a distributed network increases the availability and, this may also increase the risk of certain attacks which explore the distributed network coordination.

Security Requirements

When defining the security requirements for a WSN, the following factors should be taken in account:

- ✓ Data Confidentiality: sensor readings or other important data should not be leaked to a node's neighbors, and highly sensitive data such as keys has to go through a secure channel. Moreover, public sensor information (such as identity and public keys) must also be encrypted to protect against traffic analysis attacks.

- ✓ Data Integrity: even when data confidentially is assured, data is not completely safe because it can still be modified somewhere along its path. Therefore, data has to be checked to guarantee that it has not been modified or corrupted.

- ✓ Data Freshness: to avoid replay attacks, there is the need for a mechanism that ensures that data is recent and no old messages are being transmitted.

- ✓ Availability: additional computation and communication (for security mechanisms) consumes additional energy leading to a need of energy efficient mechanisms. Hence, a central point scheme cannot be used as it would decrease the availability and go against the concepts of WSNs.

- ✓ Self-Organization: as a WSN is typically an ad-hoc network with a dynamic topology, the common key distribution schemes cannot be applied. However, this is not mandatory and several industrial applications do not use dynamic topologies.

- ✓ Time Synchronization [20]: in order to conserve power, several schemes of turning the radio off are used. To accomplish these schemes, the nodes have to be securely synchronized with the correct time.

- ✓ Secure Localization [21][22]: localization is often needed to know a node's position. However, to avoid false reports this should be an authenticated process.

- ✓ Authentication: an enemy may not only modify a packet but also an whole packet stream. Thus, authentication mechanisms should be used to ensure that the source is correct.

Major Threats and Attacks

The main threats and attacks that can occur in WSNs can be split into six different categories.

Denial of Service (DoS) [23], a common attack to computer systems, has the objective of causing interference in the availability or performance of a given service. In WSNs, it may occur at all the layers below the Application Layer.

At the physical layer, attacks are very simple to perform. A common attack is jamming the WSN signal using a radio signal within the same frequency, which causes interference in communication and therefore integrity problems.

At the link layer, one possible attack is to intentionally violate the communication protocol (ZigBee or IEEE 802.15.4 for instance) and then send messages as an attempt to generate collisions. This is not difficult to accomplish, as it is easy to put a sniffer in the network (broadcast communication is key in WSNs) and identify the protocol by comparing to public specifications.

Also, at routing (or network) layer the attacker may use a node which takes advantage of a multi-hop network to refuse to route messages or even to route them to a wrong destination leading to losses and/or communication interruption.

Finally, at the transport layer the attacks consist basically on flooding a node with many connection requests till it runs out of resources and becomes useless. This type of attack is very common, and it is used in common networks when hackers want to shut down communication to a specific server.

Other type of attack is the Sybil attack [24][25], which can be defined as a malicious device illegitimately taking on multiple identities. With this capability, it can route multiple paths through a single malicious node and therefore obtain packets that it is not supposed to get.

However, there is another attack that is more likely to happen, simpler and more difficult to detect: node replication. This works by simply obtaining the ID of an existing node and then use it to add a malicious node which takes the place of the existing one. That existing one can either be shut down, tampered or kept to assure that the replication is not detected (if it is only for receiving).

Another important concept is privacy of data and its context. As sensor nodes collect many data, privacy is a big issue in WSNs. For instance, by knowing the correlation when analyzing innocuous data, attackers can imply very sensitive data from any environment. A good example is a WSN to monitor health of a given patient. With enough data from a sensor that measures for instance the heart rate, someone may be able to conclude that the person has a heart condition.

Traffic analysis is also a way of obtaining information to accomplish attacks or to imply sensitive data. For instance, it is possible to identify the base station (gateway) of a network even with encrypted packets, simply by knowing that nodes closer to it tend to forward more packets than the other.

By last, a factor that is not considered most times is that the deployment of WSNs is mostly done in hostile outdoor environments where there is no control or physical security. Therefore, it's easy for an attacker to tamper or destroy a sensor node with any kind of physical attacks [26].

Common Defensive Measures

To tackle the attacks and threats defined before, there are proposals that can be easily defined and implemented.

Key establishment for data communication is the most used approach to deal with security problems in general. This can be achieved using asymmetric cryptography and symmetric cryptography, depending of the scenario and resources. Asymmetric cryptography is too

computationally intensive, but feasible with the right selection of algorithms [27]. Symmetric cryptography is lighter and more used. However, the key exchange is a major problem that often dictates that asymmetric cryptography needs to be used before symmetric can be established [28].

Then, in order to avoiding DoS attacks at the different layers the following methods can be used:

- ✓ At the physical layer, one defense is to identify the jammed part of the network and route the traffic around it.
- ✓ At the link layer, nodes might use a MAC admission control which limits the communication rate and avoids traffic generated to create collisions.
- ✓ At the routing layer, the attacks can be overcome using redundancy. This is accomplished by sending a message across multiple paths.
- ✓ Finally, at the transport layer and to avoid flooding attacks, the node can ensure that the connection attempts commit the sender's resources first.

In what respects to securing broadcast and multicast communication (both inherent to WSNs), usually it is achieved ensuring that only the group members have the keys to decrypt a message. To use keys, once again the key management has to be accomplished by the group management, which can be distributed to better fit the network scheme.

In terms of routing protocols security [29], the most used technique is to add redundancy so that a packet can flow through multiple paths and therefore avoid nodes or links with security problems. Also, another technique that is directly headed to Sybil attacks is to use random key pre-distribution mechanisms, which leads to a limitation in the number of identities that a fake node can assume.

Regarding the detection of node replication attacks [30], a simple strategy is to make every node to send an authenticated broadcast message through the entire WSN. If any node receives a duplicated or conflicting claim, it revokes the associated conflicting nodes from his neighbor list.

To avoid traffic analysis attacks [31] and to maintain privacy, there are also some interesting proposals. A simple one is to randomly forward a packet to a node that it is not the source node's parent, leading to "confusion" from a node that is listening. However, to avoid time correlation attacks, a probabilistic technique is used to generate a fake packet only when its neighbor is forwarding a packet to the base station (gateway). Also, one effective measure to ensure that privacy is maintained is to flood the network with fake information among the real one or even to decrease (or vary) data forwarding within the network, which difficults the needed traffic analysis.

Finally, to deal with physical attacks there are some key measures [32]. The first, regarding the prevention of tampering, is to build a package around the node which is tamper-proof [33]. Another is to create an auto-destruction mechanism (for data and memory) when a node senses a possible attack, which also requires that parameters for attack detection are set and false positives are avoided. Lastly, it can also be used a mechanism that periodically checks if the internal memory of a sensor node has been changed. If an unprogrammed change occurs, tampering is detected and the needed measures to overcome the problem can be taken.

2.4.2. IEEE 802.15.4 Security

IEEE 802.15.4 already implements several security features [34][35], which then can be used at the MAC Layer to enable layer 2 security.

Essentially, the encryption/authentication algorithm to use is set when data is transmitted and, when data is received, the decryption/verification algorithm that has been used is defined a specific header for the receiving sensor node to be able to process it.

However, this standard as a big flaw – it does not specify how keys should be distributed, leaving this process to upper layers that can be dealt with technologies such as ZigBee.

802.15.4 Security Overview

The encryption/authentication algorithm specified in the standard is AES (Advanced Encryption Standard) with a 128 bits key length (16 Bytes). This is an important point, as having only one algorithm leads to the possibility of having hardware support for its operation, which is a big advantage to constrained devices such as sensor nodes.

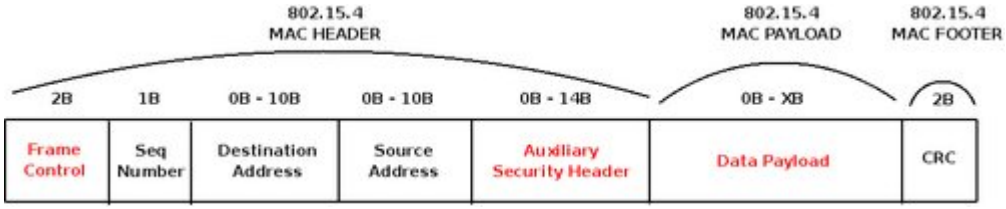

Figure 4 - IEEE 802.15.4 Frame [35]

The AES algorithm is not only used to encrypt the information but also to check and validate the data which is sent (data integrity), to authenticate nodes and to assure non-repudiation. This concept is achieved using a Message Integrity Code (MIC) which can also be named as Message Authentication Code (MAC). This code is appended to payload in the frame, and provides assurances of both the MAC header and the payload. However, it also causes some overhead to the communication that can be harmful.

The code is generated by encrypting parts of the MAC frame using a given 128 bits key that is shared by all the sensor nodes, leading to a reverse verification at the destination and therefore a comparison between embedded MIC and generated MIC using the key at destination. If the comparison is not a match, it is possible to assume that data integrity is not assured or that the sensor nodes that sent it are not part of the network (authentication concept). If different keys are used and maintained for each specific node, while verifying it is also possible to assure the non-repudiation concept (100% certain that a frame came from a given sensor node).

These MIC codes can have different sizes: 32, 64, 128 bits, however they are always created using the AES-128 algorithm with different modes of operation. Its size corresponds to the number of bits that are going to be attached to each frame, and like hashing functions the larger the number is, the more secure the verification is (less collisions). Hence, there is the need to find a tradeoff between security and the overhead that a larger size causes on the frame.

In terms of data security, it is performed encrypting the data payload field with the same 128 bits key but without generating any overhead. In fact, even the if the payload does not match

the block size of AES-128, the hardware module must support padding and therefore it deals with blocks of any size without the need to increase the size.

802.15.4 Security Details

There exist three fields in the IEEE 802.15.4 MAC frame which are related with security issues:

- ✓ Frame Control (located in the MAC Header)
- ✓ Auxiliary Security Control (in the MAC Header)
- ✓ Data Payload (in the MAC Payload field)

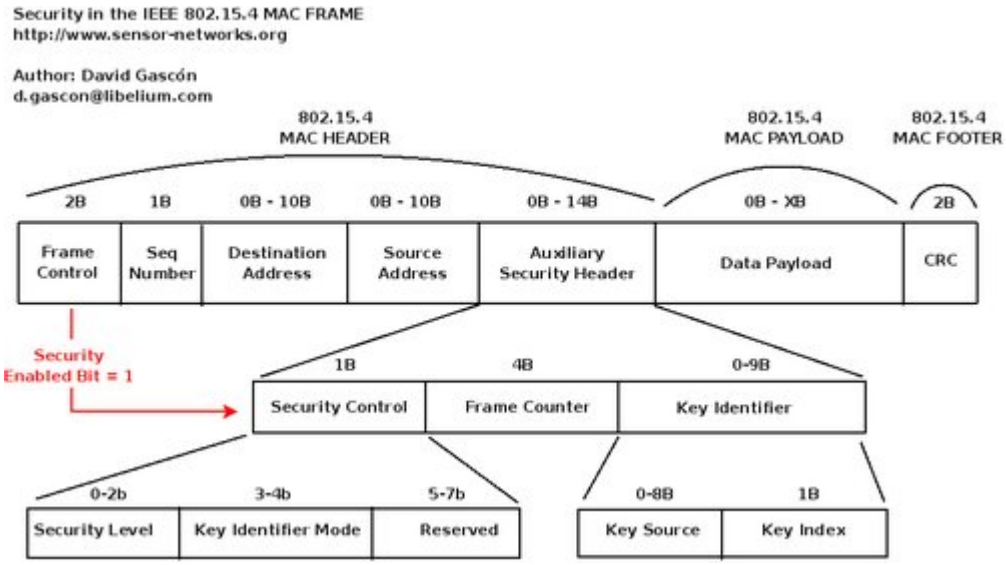

Figure 5 - IEEE 802.15.4 Security Headers [35]

The Auxiliary Security Frame will only be used if Security Enabled subfield of the Frame Control Frame is active. This particular header has 3 different fields:

- ✓ Security Control (1 byte) states which kind of protection is used (see table 1).
- ✓ Frame Counter (4 bytes) is a counter incremented by the source node to avoid replay attacks. Hence, a sequence number based on this field is used on each message.
- ✓ Key Identifier (0 to 9 bytes) allows to get information regarding the key in use to ensure both nodes are using the same.

The Security Control itself is the header where the security level is defined for the network. Using the 2 first bits (Security Level field) the AES-128 mode is chosen and also the MIC code size (if applied) is defined:

Security Level	Confidentiality	Authentication	Description
0x00	No	No	No security.
0x01	No	Yes (4 bytes)	AES-CBC-MAC-32
0x02	No	Yes (8 bytes)	AES-CBC-MAC-64
0x03	No	Yes (16 bytes)	AES-CBC-MAC-128
0x04	Yes	No	AES-CTR
0x05	Yes	Yes (4 bytes)	AES-CCM-32
0x06	Yes	Yes (8 bytes)	AES-CCM-64
0x07	Yes	Yes (16 bytes)	AES-CCM-128

Table 1 - AES-128 Modes

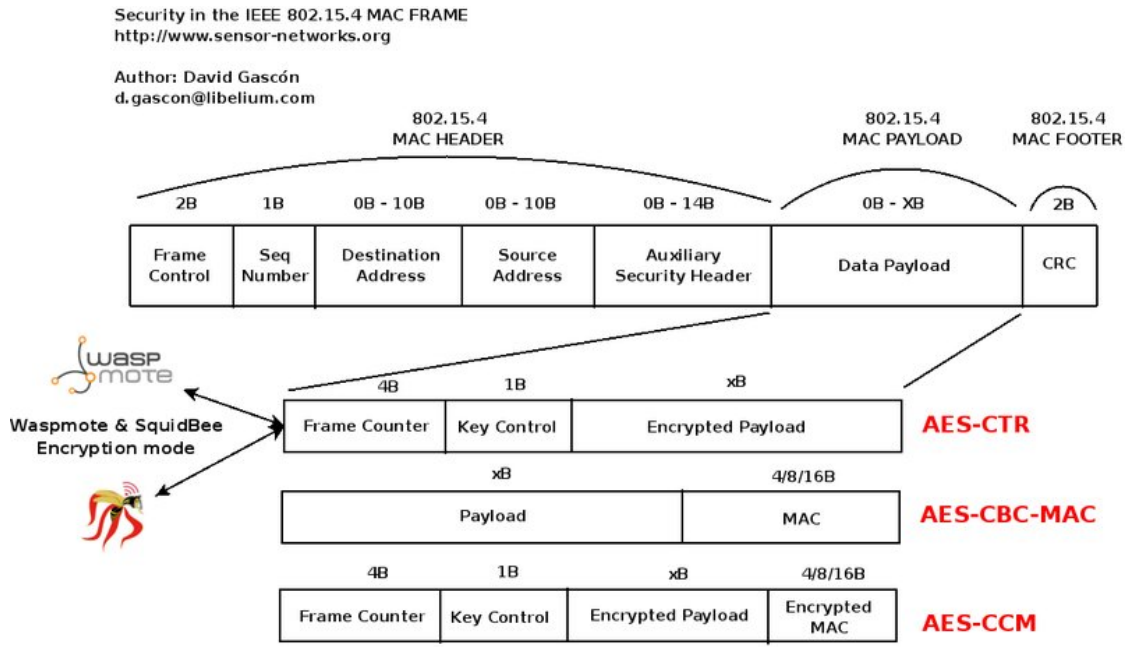

Figure 6 - IEEE 802.15.4 Security Modes [35]

The Key Identifier Mode (Fig. 5) subfield defines the type of key (implicit or explicit) which should be used by the sender and the receiver. The possible values are the following:

- ✓ 0 - the key id is agreed offline by the sender and the receiver, without any information at the message.
- ✓ 1 - the key id is obtained explicitly using the 1 Byte Key Index from the Key Identifier Field and the macDefaultKeySource.
- ✓ 2 - the key id is obtained explicitly using the 1 Byte Key Index and the 4Byte Key Source from the Key Identifier Field.
- ✓ 3 - the key id is obtained explicitly using the 1 Byte Key Index and the 8Byte Key Source from the Key Identifier Field.

As referred before, the Key Identifier field is set when the Key Identifier Mode subfield is not zero (keys agreed offline by both nodes).

The other fields are also related with keys. The Key Source subfield specifies the group Key source node while the Key Index subfield provides useful information to identify different keys from a particular Key Source.

The Data Payload field can have three different configurations which depend on the security fields configuration described above:

- ✓ AES-CTR: data is encrypted using the defined 128 bits key and the AES algorithm. The Frame Counter sets the unique message ID and the Key Counter (Key Control subfield) is used by the application layer if the Frame Counter max value enters overflow.
- ✓ AES-CBC-MAC: The Message Authenticity Code (MAC) or MIC is attached to the end of the data payload. Its length depends on the level of security specified in the Security Policy field. The MAC is generated with information both from the 802.15.4 MAC header and the data payload.
- ✓ AES-CCM: It is the combination of the previously defined methods. The subfields correspond with the AES-CTR mode while adding the extra AES-CBC-MAC subfield also with confidentiality assurances.

The Access Control List

Another important aspect of IEE 802.15.4 security is that the transceiver has to manage a list to control its trusted neighbors together with the security policy. For this reason each node has to control its own Access Control List (ACL) which stores the following fields:

- ✓ Address: address of the node to communicate with;
- ✓ Security Suite: the security police which is being used (AEC-CTR, AES-CCM-64, AES-CCM-128, etc.);
- ✓ Key: the 128 bits key used in the AES-128 algorithm;
- ✓ Last Initial Vector (IV) and Replay Counter: both correspond to the same field. The Last IV is used by the source and the Replay Counter by the destination as a message ID in order to avoid replay attacks on which an attacker attempts to send a packet that has already been transmitted.

The procedure for data transmission/reception is very simple. Essentially, the node searches the ACL to ensure the partner is a trusted node. In the case it is, the node uses data of that specific partner to apply specific security measures. In the case the node is not in the list, the message is discarded or authentication is needed to add the node to the trusted partners.

2.4.3. Zigbee Security

ZigBee [35] implements two extra security layers above IEEE 802.15.4: the Network and Application security layers. All the security policies also use the same algorithm as IEEE 802.15.4 (AES), so the embedded hardware support that is feasible for IEEE 802.15.4 may also be used. There are three types of keys:

- ✓ Master Keys: They are installed offline in each node. Their function is to keep confidential the Link Keys exchange between two nodes in the Symmetric-Key Key Exchange (SKKE).
- ✓ Link Keys: unique keys between two adjacent nodes. These keys are managed by the application level, leading to more resources utilization. Hence, they are often not used.
- ✓ Network key: It is a unique 128 bits key shared by all the devices in the network. It is generated by the Trust Center and renewed according to a defined policy. Without this key, a given node cannot join the network. This creates a mandatory requirement: the key must be pre-programmed to start the first authentication process. Every time the trust center orders a Network Key renewal, the new one is spread through the network using the old Network Key (see figure 8) to keep it safe from attackers. Additionally, as the new key starts being used, the Frame Counter parameter is initialized to zero.

Each pair of devices can have defined the Network and Link Keys. In this case, the Link key is always used which increases the security level but also uses more resources. There are two types of security policies which the Trust Center can follow according to the Zigbee standard:

Commercial mode: the Trust Center shares Master and Link Keys with any of the devices in the network according to the example in figure 7. This mode requires high memory resources (powerful devices), and therefore offers a complete centralized model for the Key Security control.

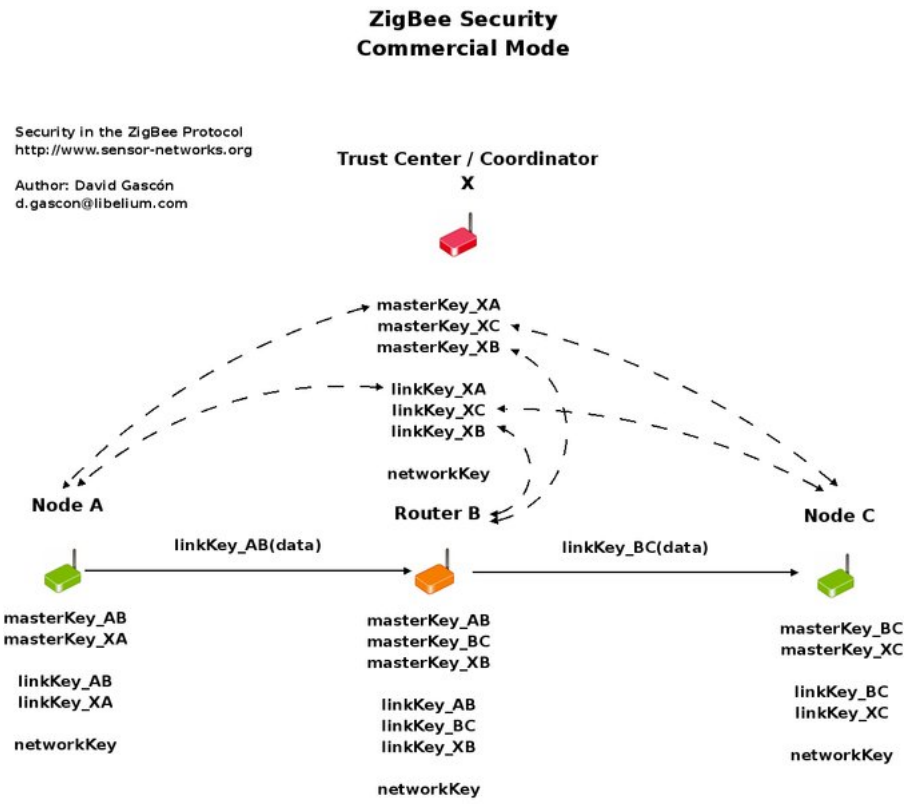

Figure 7 - ZigBee Commercial Mode Security [35]

Residential mode: the Trust Center only shares the Network Key (more useful and feasible for devices with high constrains). This is the mode that is usually chosen for the WSN topology exemplified in figure 8.

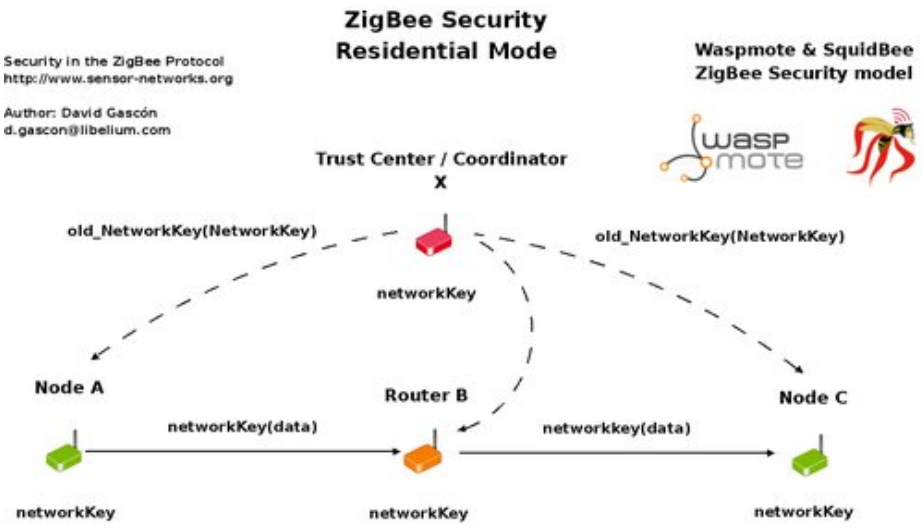

Figure 8 - ZigBee Residential Mode Security [35]

In conclusion, ZigBee provides a stable security model with some extended features that bring security to upper layers (as opposed to only IEEE 802.15.4). However, and besides the fact it is a closed and commercial technology, it does not put security below the MAC layer, does not provide any information or support regarding security levels, and it is not compatible with other communication stacks/architectures than the ones it was developed to.

2.4.4. WirelessHART Security

The WirelessHART technology [36] was designed to enable secure industrial wireless sensor network communications while ensuring ease-of-use is not compromised. In fact, its security mechanisms are so embedded in the whole technology that they cannot be disabled.

As ZigBee and 802.15.4, it also relies in AES-128 but with the concept of end-to-end sessions in order to assure that only the nodes communicating can know the contents of the payload and use that data successfully.

Risk Assessment / Reduction

According to WirelessHART's consorcium, an attacker is only relevant if it has access and enough knowledge/motivation. The security architecture embedded in WirelessHART addresses these three key points with the following actions:

- ✓ Minimize, control, and audit access.
- ✓ Require high levels of technical expertise to subvert.
- ✓ Reduce the consequences (span and duration) of any individual security breach.

WSNs security can be divided into two main categories, which can be named Data Protection and Network Protection. The first intends to maintain privacy and integrity of data transmitted along the network, while the second aims to assure the functionality of the network itself in case of internal and/or external attacks that can be provoked intentionally or not.

Data Protection

The privacy related security features implemented by WirelessHART aim to prevent eavesdropping by unauthorized devices both inside or outside the WSN. By default, a WSN with WirelessHART provides end-to-end CCM mode of AES-128 at the network/transport layer. In addition to these individual session keys, and similar to IEEE 802.15.15/ZigBee, there is a shared and common network key for all the devices in the network that eases the broadcast activity as it is needed.

The rotation of these keys (key renewal) is assured to provide an higher level of protection. However, it is not adaptable which means that they are changed automatically but according to a pre-defined security policy.

Another separate 128 bits key is also used. This is a pre-configured key intended to be used during the joining process of a node while keeping it private. Moreover, the join key also serves as authentication to the security manager that the device belongs to this network. Their policy is also different. They are treated separately from the other keys to enhance security, and can be unique to each device or common to the whole network according to the defined security policies.

Also relevant are the integrity related security features. These ensure that data sent over the WSN has not been tampered or falsified along its route. The mechanism is very simple. An integrity check field is computed and added to the packet. After the destination node receives it, it calculates the value again and by comparing to the one embedded in the packet it can deduce if the contents have not changed. In the case of WirelessHART, an advantage is that not only the payload is protected but the routing information is protected as well. Thus, attacks that aim to change routing are prevented and avoided.

Finally, data integrity also involves verifying that a certain packet has come from the correct source (non-repudiation). It works by verifying the integrity check field and the information used to generate. In fact, as the session key is unique, a certain node can verify that a given packet has came from the right sender by comparing the signature in the integrity check field (generated using that unique session key).

Network Protection

A WSN in general also needs tools to protect it against attacks. These attacks can attempt to compromise the network by inserting Trojan horse devices, impersonating networks to get sensitive data from legitimate devices, and disrupting the network to deny service. Attacks can be launched from outside or inside the company by external people or employees. Network security depends upon techniques to support Authentication, Authorization, and Attack Detection (AAA).

A WirelessHart gateway and the sensor nodes which will join its network must be configured to control which devices can access the network, because the first principle to ensure security is that all the devices respect the policies and maintain the security level. Therefore, the gateway has a defined authentication process which it uses to negotiate the join of new devices by ensuring they are legitimate and authorized to access the network. This join process, as everything in WirelessHART, is encrypted end-to-end from the node to the gateway.

Finally, to prevent DoS attacks that may try to jam the radio or overload processes such as packet acknowledgements, WirelessHART devices are capable of detecting anomalous conditions such as high levels of traffic and high number of retransmissions in order to notify an operator and solve the existing issue.

Device Roles

Devices are network routers as well as data sources, meaning that they can both forward data in a multihop topology and at the same time gather data by themselves. However, they are not authorized to play the roles attributed to the gateway as it has a specific signature that ensures that. Also important to refer is that WirelessHART does not implement the TCP/IP communication stack and therefore is explicitly safe from many typical attacks related with IP.

Security Manager

As discussed, Join, Network and Session Keys must be provided to the WirelessHART Network Manager and Join keys must be provided to Network Devices. These keys are used for device authentication and encryption of data in the network. Also as referred, the WirelessHART Security Manager is responsible for the generation, storage, and management of these keys which have different policies according to their use and the global security policy that is being enforced at the location.

There is one Security Manager associated with each WirelessHART Network. The Security Manager may be a centralized function in some plant automation networks, servicing more than just one WirelessHART Network and in some cases other networks and applications.

Transitional WirelessHART Device States:

- ✓ Idle - The device is quiescent and its wireless transceiver is not active. It has no knowledge of the WirelessHART network.
- ✓ Joining - The device is listening for the network, attempting to acquire an advertisement and requesting admission to the network.
- ✓ Quarantined - The device has successfully joined the network but only has a security clearance to talk with the Network Manager. It is not available or allowed to perform data acquisition or control functions or otherwise communicate with the Gateway.
- ✓ Operational - The device can be accessed by Host Applications via the Gateway. It is integrated in the system's operation.

Additionally, a device may also be Suspended or enter a Re-synching state after joining.

In conclusion, WirelessHART security is an industry leading technology with mechanisms that ensure that attacking a WSN with it is not easy and even if the attack is succeeded it will be minimized and detected. However, it has four major disadvantages:

- ✓ It is paid and contains a closed specification, meaning that is not the right choice for common setups and cannot be modified accordingly.
- ✓ Being closed, it is also not modular or interoperable, which can seriously affect its use across different platforms.
- ✓ It does not provide mechanisms to allow a dynamic key refresh rate based on security indicators gathered from the network.
- ✓ It lacks IP, which is simultaneously an advantage (does not have to deal with IP specific threats) but also a major disadvantage because once again interoperability and compatibility are not a reality.

2.4.5. Other Related Work

During many years, RC5 proposed by Rivest, R. and Skypjack introduced by NIST have been considered by the research community as the most relevant and suitable security algorithms for WSNs when it comes to symmetric block ciphers [37]. However, more recently technology has evolved and a new algorithm has taken its place as the most adopted and secure approach without compromising the low resource requirements of WSNs - Advanced Encryption Standard (AES) [38]. AES was initially published by Daemen, J. and Rijmen, V. as Rijndael in 1998, but with the selection process required for standardization it only came out as AES in 2001 [39]. Later, researchers have developed implementations for WSNs with optimizations and today it is considered the most preferable option [40].

Therefore, when it comes to evaluate possible security algorithms AES has to be considered as the best candidate for data protection with symmetric block ciphers, but others should not be taken apart from any comparison study. However, as AES took the main role for data protection, most of the current days embedded radios for WSNs already have its cipher implemented using hardware which leads us to great results that cannot be matched by software implementations of other algorithms in terms of speed and resources utilization.

Also, it is important to evaluate algorithms that are capable of providing integrity, non-repudiation and authentication without requiring powerful computational resources to run. In this area, industry already uses algorithms such as Message-Digest algorithm 5 (MD5) [41] and Secure Hash Algorithm 1 (SHA-1) [42] that can meet these requirements.

Regarding security algorithms comparison, there are already some proposals that try to evaluate well-known and proven implementations to general scenarios without specific requirements. However, these comparisons usually do not take in account that some platforms already have hardware based security modules and that even when a security algorithm seems to fit the platform resources it may not fit a specific application at all because of its specific requirements. For instance, Granjal, J. et al [43] evaluated algorithms such as Secure Hash Algorithm 1 (SHA1), Secure Hash Algorithm 2 (SHA2), AES and Triple Data Encryption Standard (3DES) but did not try different key sizes neither some available hardware implementations of AES available in radio transceivers such as the Chipcon CC2420 [44]. Others, like Didla, S. et al [45], extended their study past the software and also compared the hardware approach but didn't use its full potential as they let the decryption process out of the study.

Chapter 3
Proposed Model and Implementations

3.1. Overview

The proposed security model is both multi-layer and cross-layer as it can be defined as a new layer of abstraction in the communications stack. In fact, as may be observed in figure 9, it gathers and sends information to the 5 layers of the TCP/IP model with a set of modules. With this architecture, the main objective is to provide an API with direct access for the layers which can be used in a transparent way and enables security for the complete stack. At the same time, the security layer is also capable of getting information from each of the other layers and may adjust security levels according to the values of defined metrics.

This adjustment is rather important to WSNs, as depending on a whole range of factors a given node may lower the security level to save resources or to prioritize other aspects of communication.

Finally, regarding the interoperability and compatibility aspect, this model relies essentially on the use of a standardized IP stack (6lowPAN). However, other aspects are also required to assure such requirements. For instance, the code must be portable for any architecture with a C compiler, ranging from 8 bits architectures to 32 bits architectures.

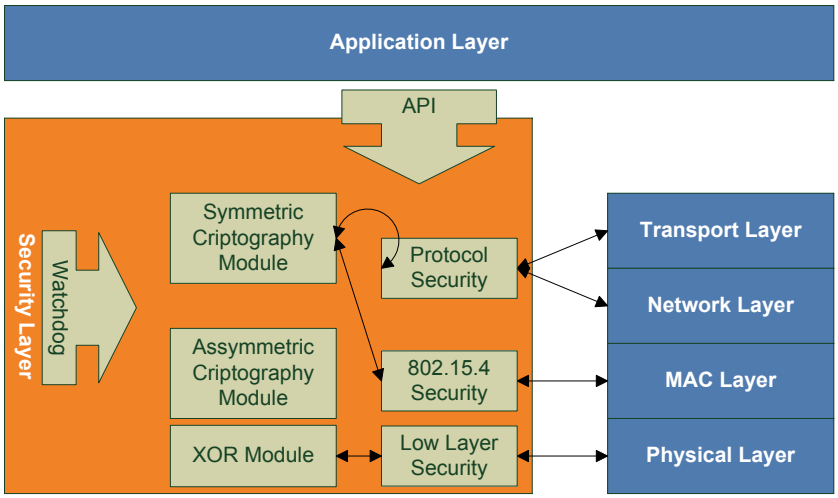

Figure 9 - Communication Layers with Security

3.2. Objectives/Requirements

The proposed model takes in account several objectives to bring innovation and a relevant scientific contribution to the existing solutions. These objectives are as follow:

- Provide a fully secure model for general use in communication stacks or standalone operations in WSNs;
- Allow the programmer to access the security layer in a transparent way through the use of a system independent API;

- Ensure that the security layer is portable for any architecture with an available C compiler (8 bits, 16 bits and 32 bits);
- Enable compatibility between different operating systems when the communication stack is the same;
- Enable interoperability with different types of sensor nodes by using a limited amount of memory, leading to a lightweight implementation (2KB of RAM and 8KB of ROM at most);
- Provide modularity to ensure that any platform is able to run the basic functions by disabling the not supported advanced features;
- Allow the definition of security levels and their dynamic change according defined metrics besides the default definitions already provided;
- Provide secure ways of distributing keys without compromising the normal operation of the sensor nodes by using efficient algorithms with optimized implementations;
- Add support for automatic key renewals based on the defined security levels or commands sent through the sink node.
- Add a contribution for the open-source community so that it has access to security mechanisms that currently are only available in closed/paid specifications.

3.3. Layers Components

Physical Layer

At the physical layer, there are not relevant proposals to provide protection against attacks such as man in the middle or denial of service. Although it may not be possible to directly protect nodes against denial of service at the physical layer, the security layer should provide mechanisms to detect the jamming and therefore allow the network layer to route the traffic through another available path.

By the contrary, to avoid attacks at the physical layer such as man in the middle, there is a simple mechanism that the security layer must implement and allow the programmer to use: introduction of low level protocol "disguise" in the signal that only other sensor nodes which are authorized can process. This is obtained at this security model through the use of the XOR module (explained later in this section). With this, it becomes much harder for an attacker to get the communication protocol and start any other type of attack. However, it adds overhead and requires more resources, which in the case of WSNs increases power consumption and becomes a drawback if the right trade-off between the security level and power is not found.

MAC Layer

At the MAC Layer, as there is the IEEE 802.15.4 open standard, this model does not bring a new implementation. However, as some operating systems (Contiki for instance) do not have support for the security mechanisms of IEEE 802.15.4 even though its set of MAC Layers is compatible with the standard, the security layer enables a whole API for the IEEE 802.15.4 security mechanisms that makes easy to add security to the already existing MAC Layers without explicit security support.

Network Layer/Transport Layer

At the network layer, this security layer provides a full set of functions to ensure that the IPv6 protocol (6lowPAN) has a tiny security header with only 2 bytes with the following contents:

- ✓ 1 byte to define how many bytes of the packet are encrypted after the security header.
- ✓ 1 byte with the following bits attribution:
  - 1 bit to select the key (key 0 or key 1);
  - 6 bits to specify the header size (if encrypted, 0 if not encrypted);
  - 1 bit for future use;

At the same time, using security algorithms for data confidentiality, data integrity, authentication and non-repudiation, it is assured that the network layer is secured and able to communicate with other network layers as long as the communication stack has the security layer defined in this proposal.

Application Layer

At this layer, as it is mostly controlled by the user, the security model only provides a set of basic functions to use in a standalone way in order to do operations which may not even be related with communication. These functions can be accessed the same way the communication layers do – an API. This API is described in Annex A of this dissertation.

3.4. Layers Communication

One key aspect when designing any security model is the communication between layers in a communication stack. In fact, security can be classified as end-to-end security or hop-by-hop security. In the first case, secure communication is established between two points and other nodes which relay the messages should not be aware of their content. In the last case, it is the opposite as every node in the path from source to destination should be able to decrypt the contents of the message.

Bearing this in mind, the security model proposed in this thesis provides a hybrid approach as other relevant proposals described in the state of the art do. This approach can be viewed in Figure 10.

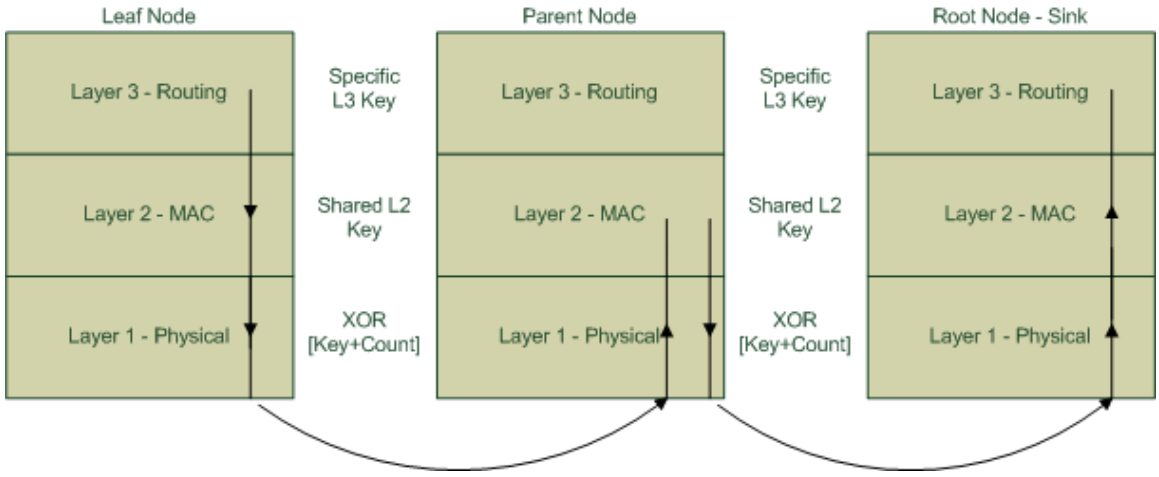

Figure 10 - Multihop Secure Communication

In this figure, it is considered that the leaf node is sending data to the root node (also known as sink node). However, as there is no direct path, data has to pass through the parent node so that it may reach the destination (typical multihop scenario). To achieve this communication with security, data is encrypted/signed with a specific layer 3 key established between the leaf node and the root node (only these nodes can get the decrypted data) and also with a shared layer 2 key that is generic for every node in the network (enables routing as frame and packet headers need to be decrypted). Additionally, a XOR process based on the layer 2 shared key is also done at the physical layer to protect the communication protocols. The arrows in the picture describe this process, but only as far as the security model goes (unsecure/application related security are not shown).

3.5. Symmetric-key Cryptography

Symmetric-key cryptography is a class of algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. The keys can be the same or there should be a simple process to achieve one key from the other. These keys basically represent a shared secrete between two nodes that is used to keep a "tunnel" with private information.

Symmetric-key cryptography is also known for not being computationally intensive in general, but when comparing to asymmetric-key cryptography it has a major drawback: keys have to be shared between the nodes. This leads to some security problems and creates the need for an effective key exchange mechanism.

Security Ciphers/Hashing Functions Evaluation

In order to evaluate possible ciphers and algorithms implementations that can be used for the model proposed in this work, the following algorithms/hashing functions have been evaluated:

- ✓ AES-128: AES is a symmetric key encryption algorithm, which uses keys of 128, 192 or 256 bits. However, only the 128 bits version was evaluated as the hardware implementation is AES-128 and does not support longer keys. The block size is 128 bits.

- ✓ 3DES: this algorithm is also a symmetric key encryption algorithm. However, it relies on simple DES applied 3 consecutive times, each with a different key and varying the operation (encryption, decryption and encryption once again). These keys can have 56 bits, 112 bits or 168 bits and the block size is 64 bits.

- ✓ MD5: Message-Digest Algorithm 5 (MD5) is a widely used cryptography hashing function that given a certain input generates a 128 bits digest that enables the verification of a message after it has been transmitted over a WSN and that also can provide authentication.

- ✓ SHA1: SHA1 is another cryptographic hashing function that generates a digest of 160 bits for a given input. Unlike MD5, the larger digest reduces the probability of hashing collisions.

In order to run a conclusive and valid study, tests specification played a main role.

Firstly, the traffic sources running in each node had to be decided. In this particular case, nodes ran a simple application that gathered data from sensors (temperature, humidity, etc.) at a given rate and sent it through a single stream to the sink node (multi-hop scenario). The rate of sampling from the sensors was also the rate at which the nodes were sending data, and in this case it made sense to test 0.25 Hz, 0.5 Hz and 1 Hz. Also, the packet payload size varied during the tests, being 16 bytes or 32 bytes.

To obtain the results presented below, raw measures of time and power consumption were obtained. To obtain time, the hardware clock in TelosB motes was used. This hardware clock has a frequency of 32768 Hz which leads to a precision of 1/32768 seconds. Finally, to obtain power consumption results, a software module named Energest (included in Contiki operating system) was used. It measures the time used by each hardware component and according to their datasheets specification obtains the power consumption in one hour (microwatts/hour). Also, tests were conducted for 10 minutes to let the network stabilize and to obtain a representative sample that ensures a good confidence interval to the statistical averages.

In the specific case of AES-128 supported by hardware, no implementations were found across the research community that contemplate both encryption and decryption (or decryption only) for high level access (layers above MAC Layer such as the network layer). This is mostly a cause of the CC2420 datasheet statements, which redundantly state that only encryption is supported for direct access (standalone mode). However, as some authors such as Didla et al. speculated in their algorithms comparison work, there should be the possibility of doing both operations if the process of sending and receiving data is emulated (in-line mode is supported with both operations for MAC layer security). This has been explored in this work and after big efforts to circumvent the limitations (eg. RX buffer is not

writable unless a trick is applied), lack of documentation and dozens of tests, a full implementation was achieved with excellent results and a lightweight code base. This implementation uses the following steps to provide both encryption and decryption:

- ✓ Read security register 0.
- ✓ Ensure that no key is set and that in-line security is disabled in register 0 by using bit masks.
- ✓ Select key 0 or key 1 through register 0 (already in memory) for the encryption/decryption operation.
- ✓ Set in-line security mode to AES Counter Mode (AES-CTR) in register 0 to enable encryption and decryption operations.
- ✓ Write security register 0.
- ✓ Set nonce for the counter block in big endian byte order.
- ✓ Read security register 1.
- ✓ Set TX/RX (encryption/decryption, respectively) frame header size to 0 in security register 1 as we are not interested in using complete frames but rather to use our own payload/data.
- ✓ Write security register 1.
- ✓ Write data to encrypt/decrypt to the TX/RX FIFO queue depending if the operation is encryption or decryption.
- ✓ Send strobe to encrypt/decrypt TX/RX FIFO queue contents.
- ✓ Wait until the operation is complete.
- ✓ Read total length of data in TX/RX FIFO queue.
- ✓ Remove automatically added footer from length (discard).
- ✓ Read encrypted/decrypted data from TX/RX FIFO queue.
- ✓ Restore registers 0 and 1 to defaults.

Confidentiality Assurance

The first test is related with the encryption/decryption algorithms, comparing the time and power spent by each of them to encrypt/decrypt 16 bytes of data. Data regarding this test is presented in bar graphs in figures 11 and 12.

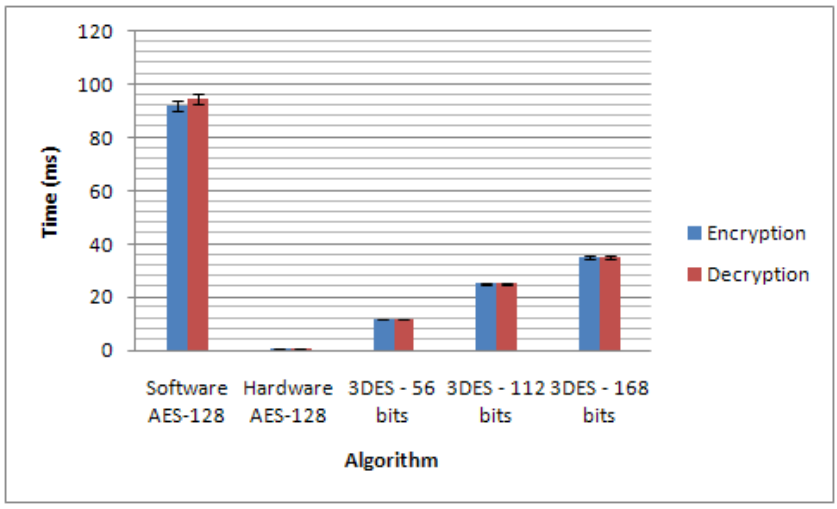

Figure 11 - Encryption and decryption running times with 16 bytes payloads

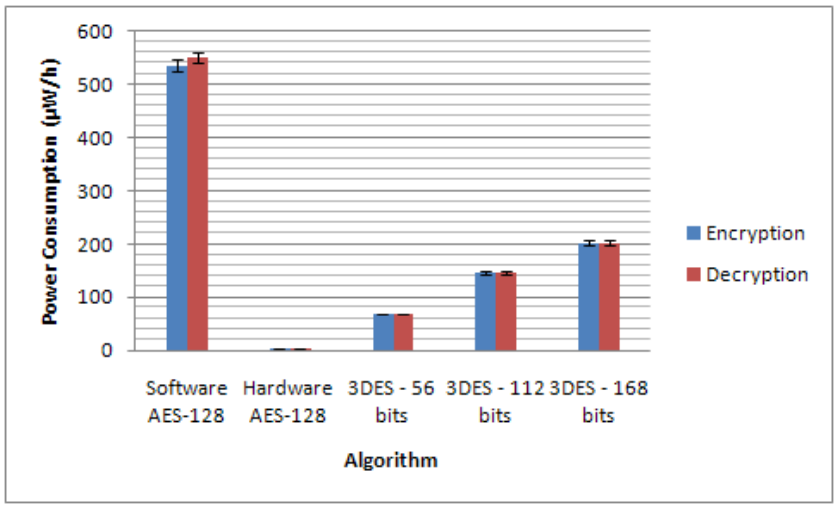

Figure 12 - Encryption and decryption power consumptions with 16 bytes payloads

The second test it is almost like the first but uses 32 bytes payloads as reference, being represented in figures 13 and 14 for both running times and power consumptions, respectively.

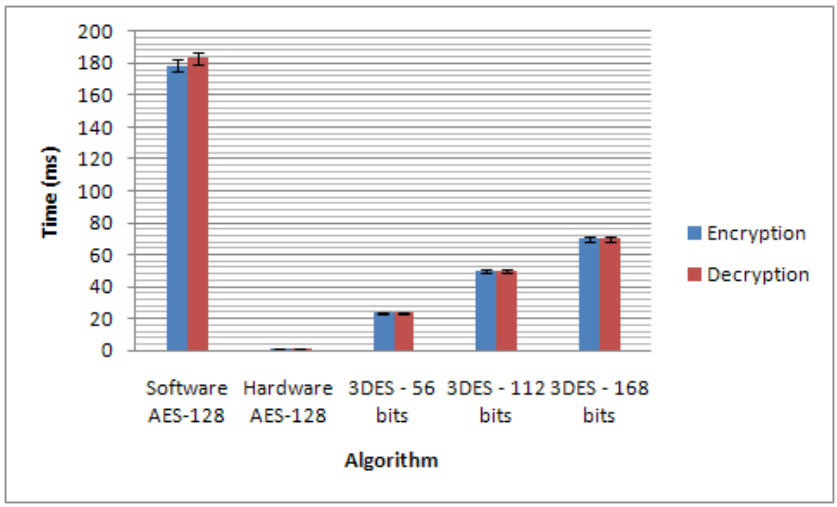

Figure 13 - Encryption and decryption running times with 32 bytes payloads

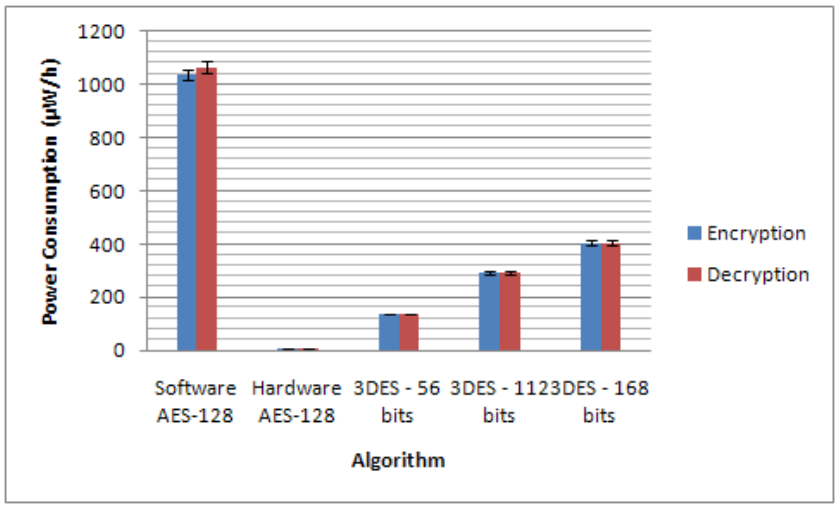

Figure 14 - Encryption and decryption power consumptions with 32 bytes payloads

As illustrated in the above graphs, most algorithms take a long time to run. Specially, software AES-128 wastes almost 100 miliseconds to encrypt/decrypt data, turning it a real bad choice to use in real-time environments. Also, regarding the difference between encrypting and decrypting, only software AES-128 takes different times to do both operations. From this, one may conclude that for a valid proposal of a security model than can be interoperable, software algorithms are not an option.

Lastly we may also observe that using 32 bytes payloads duplicates the time and energy consumption in all the cases. Thus, the conclusions taken with 16 bytes payloads remain valid and become even more supported.

Regarding energy, as it is correlated to time because it was measured with software, the same conclusions can be taken as the ones referred above. There is only one special remark: hardware AES-128 does not use CPU power for its execution, meaning that the results presented here only take in account the processing at the application level. However, according to CC2420 datasheet the energy used by the AES-128 hardware module is so unrepresentative that it may be discarded in evaluation studies.

Bearing all these conclusions in mind, the choice would obviously go to the hardware based AES-128. However, if there is no hardware support, there should be used a software algorithm. In this case, it is a hard choice. Altough 3DES with a small key could be a logic choice, it is not very secure due to its inherent characteristics. This led to another approach: optimize the AES-128 software algorithm. In fact, Brian Gladman [46] developed a version with such optimizations that the performance can increase to a point where the algorithm runs in a time under 1-2 milliseconds as supported by Didla, S. et al [45]. This meets the requirements of this security model, leading to a double option for confidentially assurance in this model: AES-128 hardware and AES-128 software.

Integrity, authentication and non-repudiation

Other relevant tests were the ones regarding cryptographic hashing algorithms. Below, some graphs present the collected data in figures 15 and 16.

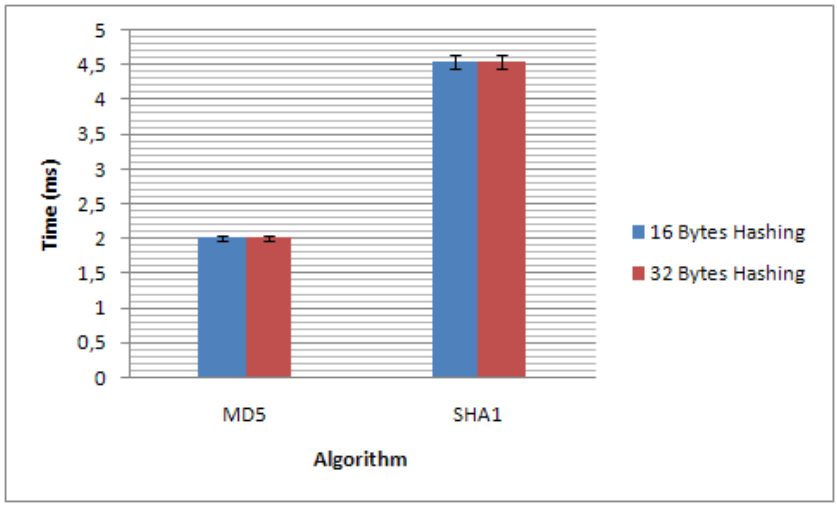

Figure 15 - Cryptographic hashing algorithms running times

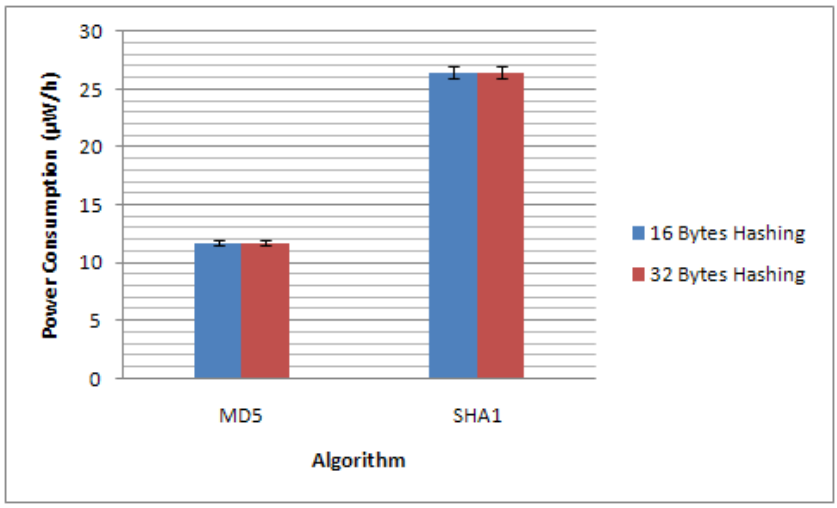

Figure 16 - Cryptographic hashing algorithms power consumptions

From the graphs, and taking in account that the model related with this work intends to have low power consumption and low execution time, the first conclusion is that only MD5 may be applied. In fact, SHA1 takes twice the time and energy to run and generate a digest that is not twice as big. Thus, SHA1 is not a valid option.

Also, an interesting fact is that both algorithms take exactly the same time and energy to create digests for either 16 bytes or 32 bytes payloads.

Finally, as this model needs these mechanisms, the option for a software based algorithm fell on MD5. As hardware support is concerned, AES-128 with CBC mode or CCM mode also deal with the requirements for integrity, authentication and non-repudiation.

3.6. Asymmetric Cryptography

Asymmetric cryptography refers to a system that requires two separate keys: one secrete key and one public key. There can be considered three types of asymmetric cryptography systems: public key cryptosystems, public key distribution systems and digital signature systems.

The first one allows secure communication between two nodes using algorithms such as Ron Rivest, Adi Shamir and Leonard Adleman algorithm (RSA)

The second one allows two nodes to securely agree on a shared secret, even if all their communications links are compromised. The shared secret generated from this process is then typically used as the key in a symmetric cryptography system.

Finally, the Digital Signature Algorithm (DSA) is the most widely used digital signature system. It can generate a digital signature, but has no privacy features and only enables the verification of a given entity.

As explained, these algorithms rely on a key pair (public and private key). These keys are mathematically linked, and the private key is extremely difficult to obtain from the public key. This is due to the mathematical relationships involved (the most notable ones being the integer factorization and discrete logarithm problems) that have no efficient solution.

Its process is relatively simple. The public key is used to transform the message into a form that cannot be decryptable without the matching private key. Therefore, any node which advertises its public key enables any other node to produce messages that can only be read by it because it is the only one with the private key.

However, asymmetric cryptography has a major drawback: it is very computationally intensive. Considering this issue, this model will only apply it for key distribution. After the key exchange process, the less resource intensive symmetric cryptography will be used.

3.6.1. Key Distribution

Key distribution is a fundamental topic when defining a security model than can provide a high level of security together with low resources utilization. A well known method to establish keys securely is the use of the above described asymmetric cryptography, only taking in account that asymmetric cryptography tends to be resource intensive.

However, if there is the requirement that only certain nodes may join a given network and start the key exchange procedure, other security measures should be involved. These are shown in Figure 17.

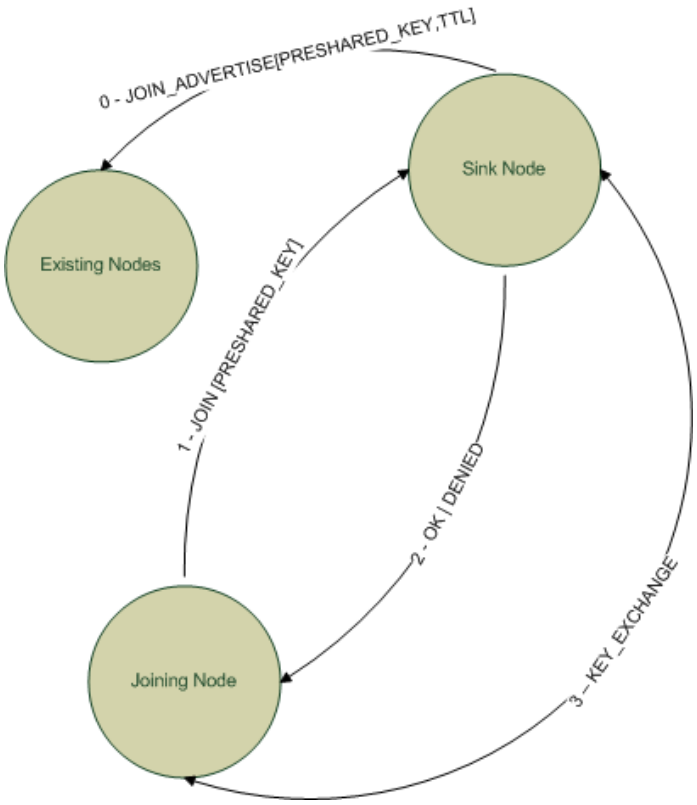

Figure 17 – Secure Join Mechanism

First, the sink node (connected to a computer) sends a secure layer 2 broadcast message advertising all the nodes in the network that a node with a given pre-shared key may join the referred network. This key as a Time To Live (TTL), meaning that for security purposes it will only be valid for a defined time according to the specific scenario.

Then, when the joining node wants to join the network, it uses a ROM burned pre-shared key to issue a join request (layer 2 communication with shared key security) to the sink node (through other nodes if multihop is used). If the join request is approved, the key exchange then starts so that the node receives the network layer 2 key and any specific layer 3 keys used to end-to-end communication.

For evaluation purposes, and during a collaboration meeting with the University of Murcia from November 19th to November 30th, a simple well known and proven algorithm that deals with key distribution was tested and adapted to WSNs – Diffie-Hellman [47].

3.6.2. Diffie-Hellman

Diffie-Hellman was first published by Whitfield Diffie and Martin Hellman in 1976, and today it is still used as the base for public-key cryptography. However, and even though its operation is simple (as it may be observed in figure 18), it as a major drawback when applied to constrained devices such as sensor nodes: it needs to deal with really big numbers in order to be secure. Therefore, our attempt to implement and run it on a 16 bits device was not successful at first, but with some mathematical simplifications and the use of binary operations it was possible to achieve a functioning implementation. For instance, with the

biggest number the architecture supports without extensions (2^16) it was possible to establish a key pair in 26 milliseconds.

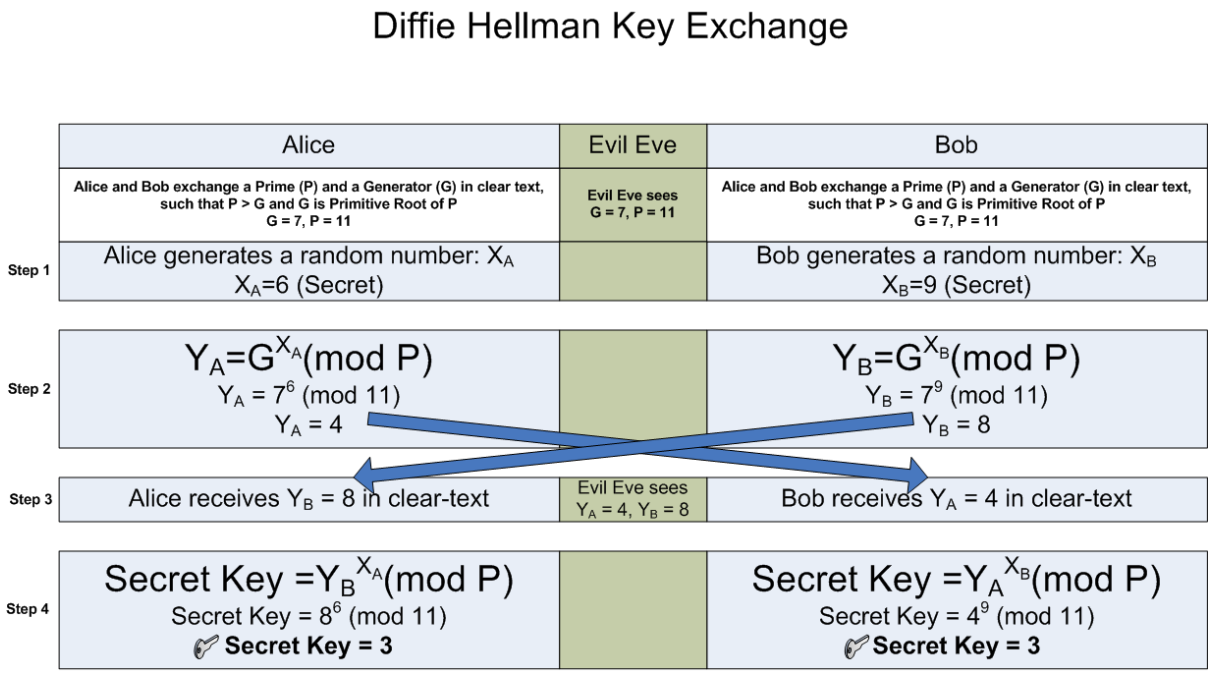

Figure 18 - Diffie-Hellman [48]

Although the results may look good and 26 milliseconds is indeed a good execution time, numbers in the order of 2^16 are still too low to ensure that man in the middle attacks (security threat on which the attacker listens to traffic between 2 points) are avoided. But, as the architecture does not support bigger numbers without extensions and even with those extensions it would still be computationally too intensive, it became a problem to find an effective and secure way of distributing keys. As a possible solution to be explored further in this work, it was found Elliptic Curve Diffie Hellman (EC-DH), which according to some proposals [49][50][51] may be modified to fit the resources of even 8 bits architectures.

3.6.3. Elliptic Curve Diffie-Hellman

Elliptic Curve Diffie-Hellman (EC-DH) is a cryptosystem which shares same ideology behind Diffie-Hellman but uses an elliptic curve as base. An elliptic curve is a plane curve which consists of the points that satisfy the equation

$$y^2 = x^3 + ax + b$$

along with a distinguished point at infinity, denoted ∞.

An example of a simple elliptic curve may be found in the next figure:

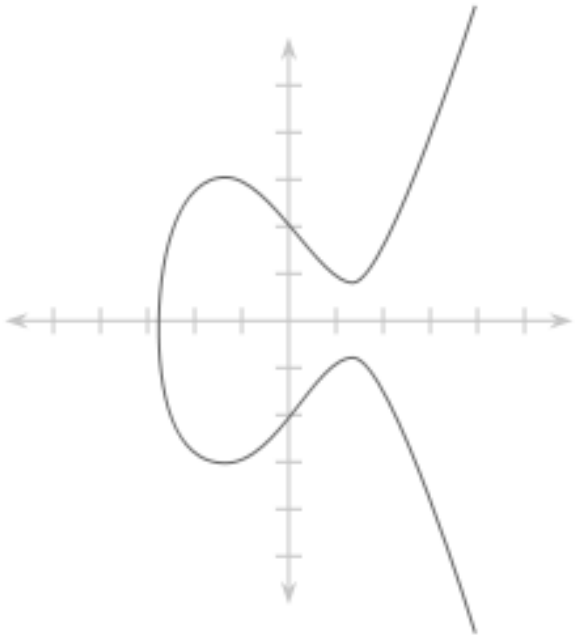

Figure 19 - Elliptic Curve Example [52]

As other asymmetric cryptography systems, it is based on the intractability of mathematical problems. In this case, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible. This was first suggested by Neal Koblitz[53] and Victor S. Miller[54] in 1985.

Considering this, Elliptic Curve Diffie-Hellman is considered more secure than the common Diffie-Hellman by NIST [55]. In fact, it requires a shorter key to deliver the same security performance as Diffie-Hellman. NIST also states that that EC-DH key should be twice the length of equivalent strength symmetric key algorithms as the following table demonstrates:

Symmetric Key Size (bits)	Diffie-Hellman Key Size (bits)	EC-DH Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Table 2 – NIST Recommended Key Sizes

But security is not the only attractive feature of elliptic curve cryptography. Elliptic curve cryptosystems also are more computationally efficient than the first generation public key systems such as RSA and Diffie-Hellman. Although elliptic curve arithmetic is slightly more complex per bit than either RSA or DH arithmetic, the added strength per bit compensates

any extra computing time. The following table shows the ratio of DH computation versus EC-DH computation for each of the key sizes listed in table 2:

Symmetric Level (bits)	Ratio
80	3:1
112	6:1
128	10:1
192	32:1
256	64:1

Table 3 - Relative Computation Costs of DH and EC-DH

As far as key exchange is concerned, the algorithm is identical as already referred. It may be defined by the following steps (considering the example in figure 18):

- ✓ Alice and Bob agree on a elliptic curve, its underlying field and the base point.
- ✓ Alice and Bob generate a random value in the range $(1, r-1)$ where r is the order of the elliptic curve. This value will be the secret key of each of them.
- ✓ Then, they both calculate their public key by multiplying their secret keys times the chosen point on the curve.
- ✓ After that, they both exchange their public keys. After receiving the public key of the other partner, each of them multiply it with their secrete key. They both obtain the same number, which will be the key for symmetric cryptography.

However, this poses some problems for architectures with 8 or 16 bits because it keeps using large numbers. This is an issue that has to be solved and admitted, an can be overcome through the use of extensions that represent big numbers using 8 or 16 bits words and use carries to implement the arithmetic operations.

Even though EC-DH still uses big numbers and complex operations, the following table illustrates the differences between implementations of DH and EC-DH in TelosB motes. For comparison purposes, DH with 3072 bits and EC-DH with 256 bits have been used. The primes and curves are the ones suggested by NIST for testing purposes. To obtain statistically relevant data, 30 samples were collected at each node and the presented values are the average of those samples with a standard deviation of 5%. Time and energy values were measured using the mote's internal clock and the Energest (energy measuring) module of Contiki operating system. RAM and ROM values have been obtained with the tool msp430-size (included in the MSPGCC compiler) and represent accurate values measured only once.

Metric	Diffie-Hellman	EC-DH
ROM (bytes)	2902	3420
RAM (bytes)	458	635
Time on each node (seconds)	100,58	1,6
Power Consumption on each node (µW/h)	50274	802

Table 4 - DH versus EC-DH

Considering this results, EC-DH was chosen as the key distribution algorithm for the security model proposed in this thesis. Additionally, its measured performance was improved with a set of optimizations based on observation and study from the author of this dissertation:

- ✓ Assembly Coding: parts of the code were rewritten to Assembly in order to save cycles introduced by inefficiencies of the C compiler.
- ✓ Memory Usage Reduction: usage of RAM was decreased by eliminating unneeded variables and debug outputs.
- ✓ Optimize Arithmetic Operations: representation of big numbers with parts that were 0 have been removed from the operations.
- ✓ Pre-calculated values: in some cases where communication always happens with the same node (sink node), some values may be pre-calculated. This means that the agreement on the curve and the base point is established at programming time and do not have to be agreed and communicated at the key exchange process.

3.7. XOR Module

Some industry leading standards such as WirelessHart and Zigbee do not provide mechanisms to enable security at the physical layer of the communication stack. This makes them vulnerable to protocol violations and man in the middle attacks as explained before. In order to prevent most of these attacks, a module was defined at the proposed security model.

This module is named XOR as the main operation it uses is an exclusive OR, a very efficient computing operation. Essentially, it uses one byte of the layer 2 shared key and RX/TX counters to produce the number that will be used to XOR the bytes that are sent or received. As XOR is an operation that is reversible doing exactly the same, both the sender and the receiver perform the same operation with data they both share. Without knowing

the number and considering that it will change every time, the attacker will not decrypt the protocol that is being used. This will put some extra effort on breaking in attempts while using low computational resources and a small amount of memory ($N * N\ bytes$) where N is the number of nodes in the network/the number of nodes communicating.

The process that nodes use is the following:

- ✓ The sender (x) takes the number of messages that it has sent to the receiver (y) given by $N(x, y)$.
- ✓ It then computes every byte at the TX buffer (B), excluding the field with the number of bytes and a special field with the sender ID (at the beginning of the buffer), with the formula

$$B\ XOR\ (KEY[N\ mod\ 16] +\ N\ mod\ 256)$$

  where B KEY is the layer 2 key (16 bytes) shared by every node.

- ✓ Then it sends the contents of the TX buffer to the other node.
- ✓ After the receiver gets the message, it takes the $N(x, y)$ where x is the sender and y itself (the receiver) and which represents the number of traded messages between the two nodes and computes the same XOR operation with the same formula explained above.
- ✓ Finally, the receiver has the translated message.

3.8. Watchdog

With dynamic security levels requirements and attack detection needs, a watchdog running at the security layer is a logical decision for a module addition.

Mainly, it has the following responsibilities:

- ✓ Keep track of keys at memory, which are stored at RAM for software implementations and at the radio transceiver buffers for hardware implementations. If there is an order to change keys from the Key Distribution Center (KDC) which is the sink node, the watchdog does that change and notifies the involved modules;
- ✓ Maintain pre-shared keys register and track their TTL to allow their expiration as programmed;
- ✓ Maintain a register of anomalous situations that may indicate attacks;
- ✓ Monitor resources such as battery to act accordingly;

- ✓ Share general information between the modules.

The following table illustrates the occurrences and the actions that the watchdog should as its default behavior:

Occurrence	Action
Battery Drops 5%	Lower the security level until battery is changed
Resources Occupation is High (> 90%)	Lower the security level for 10 minutes
KDC Orders Key Renewal	Notify involved modules to renew key and stop forwarding data while not renewed
Pre-shared Key of Other Node Expires	Notify MAC layer and symmetric cryptography module
Order to Change Security Level	Change security level
Abnormal Traffic	Notify the sink and wait for orders
High Number of Integrity Verification Fails	Notify the sink and wait for orders
Order to Ignore Compromised Node	Notify routing layer to re-route traffic

Table 5 – Watchdog Actions

These occurrences and their corresponding actions have been decided as the default behavior for the watchdog after tests and observation. In fact, the first two occurrences state specific values that could not be used without proper testing.

Regarding the battery, the 5% value was observed in tests that were used for autonomy validation and which are specified in the next chapter. The value is related with the boundaries of battery levels on which motes can operate.

In terms of resources occupation, the occurrence of more than 90% of resources in use and the action of lowering the security level for 10 minutes have been decided after testing a network with a high load, mostly because it was the combination that offered a better tradeoff between security not being too low while saving a good level of resources to accommodate peaks of high load.

3.9. Dynamic Security Levels

When developing any security system, a key point to focus on is the definition of security levels. With such definition it is possible to adjust the level of security according to the system's conditions or pre-defined policies.

For instance, a given network may not need confidentiality or it may not be a major concern, while for other it may be critical. Also, depending on the platform, there may be factors which become more important than security and cause the change for a lower security level to save resources dynamically.

Considering this, a set of basic security levels which can be changed dynamically by the Watchdog has been defined in table 4. However, the API allows the definition of custom levels so that they may be adjusted to specific scenarios.

Level	Key Distribution	Layer 1	Layer2	Layer3
Extreme	EC-DH 256 bits	XOR	AES-128-CCM	AES-128 (Full)
High	EC-DH 256 bits	XOR	AES-128-CCM	AES-128 (Auth Only)
Medium	EC-DH 256 bits	XOR	AES-128-CCM	-
Low	EC-DH 256 bits	XOR	AES-128-CTR	-
Very Low	EC-DH 256 bits	XOR	-	-

Table 6 – Default Security Levels

The table above represents the five security levels defined by default in the proposed security model. Key distribution is fundamental for the node to get key renewals, while layer 1 security with the XOR module is the most basic security that can be applied. Therefore, these two modules are the basis for all the default security levels.

However, the same does not apply to other modules. At layer 2, there are multiple combinations defined by IEEE 802.15.4. In this case, it only makes sense to use CCM (both integrity/authentication and confidentiality) or CTR (confidentiality), bearing in mind that the first uses twice the resources of the second and adds overhead (MIC) to the frame. Therefore, when optimizations are not possible at layer 3, adjust should be done to use CTR only. In alternative, CBC can also be configured if the focus of security is more on integrity than confidentiality.

Finally, at layer 3 there is also the possibility of using AES as encryption and integrity/authentication mechanism or solely as integrity/authentication algorithm. Even though AES-128 is referred in the table, for authentication/integrity the algorithm in use may be different if hardware AES-128 is not supported. In the case of this model, MD5 should be the mechanism in use.

Chapter 4
Proposed Model Evaluation

4.1. Tests Specification

In order to validate the proposed model, a number of tests are defined. The first category of tests involves resources utilization, and aimed to evaluate the performance impact of the security model in industrial scenarios using a case study described below. The second category involved integration tests, and aimed to evaluate the compatibility, interoperability and portability of the model.

The first set of tests is aimed at the first category. Essentially, a number of metrics which are relevant for WSNs are defined and explained in the next section. These have been evaluated in the case study scenarios with its architecture with assurance of statistical representation by running the network for 24 hours and collecting huge samples that then were processed to obtain averages and standard deviations.

The last set of tests involved 2 motes: a TelosB (16 bits architecture) running TinyOS and a Jennic JNB5139 [56] (32 bits architecture) running Contiki operating system. Both had the proposed security model implemented, and were set to communicate with 6LoWPAN sending the environment temperature every 3 seconds. A successful communication link and operation for 24 hours was defined as the objective to conclude that the security model integrates well at any architecture, with any protocol at most operating systems.

4.2. Evaluated Metrics

As defined at section 2.3.1, there are many obstacles to apply security to WSNs. These range from the limitation of resources to their wireless communication limitations. Therefore, a validation of any security model should pass at first level for the evaluation of metrics regarding these limitations.

4.2.1. Battery Life

Before the presentation of the evaluation studies, it is important to understand the consumption properties of the batteries used in the TelosB nodes. This knowledge will be useful to fully interpret the battery consumption evaluations.

The main restriction in the WSNs components, the nodes, it is the limited access to power supplies. Most nodes use simple AA batteries, which have a very particular discharge process.

A study performed over MSP430 hardware [57] concluded that in the beginning of the lifetime, the battery energy decreases slightly faster than past its middle. Once in the middle, the battery consumption is constant. The next figure shows the curve of the battery discharging achieved by Kramer et al:

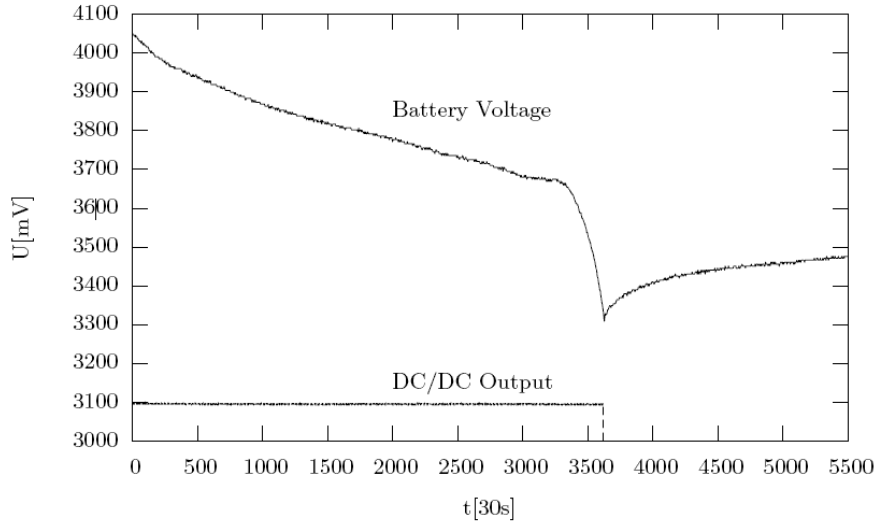

Figure 20 – Battery Consumption Example [57]

In real scenarios, these characteristics are common and not only when the battery is new. Every time the node is turned on, the battery requires a fixed time to become stable, achieving its real value. In the study it takes about 250x30 seconds for the battery to stabilize.

For instance, the battery consumptions in TelosB motes (according to its datasheet) are reported in samples defined by the follow expression:

$$Vbat = \frac{Vref\ x\ ADC\_FS}{ADC\_count}$$

Where: Vbat = Battery Voltage, Vref = Internal Voltage reference = 1233mV, ADC_FS=1024, and ADC_count is equal to measured value.

As there is a conversion from an analog value to a digital value through an analog to digital converter (ADC), the sampling rate causes some loss of information.

If the measured value (ADC_count) was 475 (decimal units converted from the hexadecimal read value), the Vbat would be equal to 2636,5305. The next measured value would be 476, which means a Vbat equal to 2630,9916. Hence, the reported battery value never decreases in one by one unit. Instead, it decreases in samples due to the TelosB conversion formula (for instance 6 in 6 mV). This characteristic is very important to understand the evaluations presented ahead in this thesis.

4.2.2. Performance and Resources Impact

In order to evaluate performance and resources impact over a network setup in a critical environment such as the one described in the next section, a number of metrics have been defined. The first one was latency.

On specific scenarios, it is critical that the delay between a given event and its acknowledgement at the destination is deterministic and below a certain threshold. Therefore, any security mechanism that runs continuously cannot have a great impact on the delay by taking too much time to run. To measure the end-to-end latency, clock synchronization was achieved through a specific module in Contiki operating system (nodes set their time based on messages from the master/sink node) and a timestamp at each packet provided the value to obtain the delay time. It was obtained with a precision of 1/32768, which corresponds to the maximum frequency of the hardware clock (32 MhZ) of the chosen platform.

The other two metrics were related with resources utilization. As multi-threading is not used in the test scenario, CPU was not considered relevant. However, even in other scenarios, while running security algorithms there should not be more resource intensive computation occurring and CPU would not be a problem. Hence, for this evaluation study the only metrics considered were the utilization of RAM and ROM. ROM is easy to evaluate, as the size of the binary that is programmed to the mote corresponds to its utilization. However, RAM utilization measurement was not immediate. To obtain its value, one needs to know that typically memory allocation is not dynamic. As it is static, there is a small tool included with MSPGCC compiler that gives the value of bytes allocated at RAM (msp430-size). With this tool, accurate results have been obtained.

4.3. Case Study - ICT FP7 GINSENG

4.3.1. Overview

ICT FP7 GINSENG – Performance Control in Wireless Sensor Networks [14], in a brief summary, aims to create WSNs that meet demanding performance specifications and that can be applied in industrial environments where real-time operation is critical.

Nowadays, WSNs allow the monitoring of several factors in a huge number of different scenarios. However, most deployment scenarios are not critical and thus become tolerant to packet loss, communication failures, delays and unreliability in general. However, there a few critical scenarios where the requirements are strict and the tolerance to any kind of failures or bad performance are minimum. Bearing this in mind, GINSENG aims to develop a complete WSN system that can be deployed in such environments and therefore meets the requirements. At the same time, such a system would bring several advantages such as: fast and easy deployment, low costs of deployment/maintenance, easy adaption to any scenario and flexibility. These advantages get even more important if a WSN system is compared with current systems, which demand the use of cables and have huge costs of installation and maintenance.

Finally, in order to provide an industrial prototype that can evaluate the system in a real world scenario, one of the partners is Petrogal, SA. This company has two oil refineries, but for the case of GINSENG only the one in Sines will be considered.

Using this oil refinery as a deployment scenario, GINSENG is able to test the monitoring and actuation using WSNs at a real scenario with strict requirements in terms of performance and reliability. In fact, the administration has established some key points for performance:

- ✓ In most scenarios the delay has to be up to 1 second;
- ✓ Packet loss cannot be higher that 0.1%;
- ✓ Packets generated from alarms cannot be lost at all;
- ✓ Orders to critical actuators have the same requirements of packets generated from alarms, although there is some flexibility.

As referred in the introduction, the author of this dissertation participated in GINSENG project. In fact, he took an active role in the team of researchers that elaborated the proposals and evaluation studies for the modules, which now are the core of the project. This is an important factor for the inclusion of the project in this report, mostly because being deeply inside of the project's conception allows the proposal of a security model with a higher level of knowledge that in other conditions would not be possible. This, combined with the strict requirements inherent to the environments where GINSENG is applied, makes the project the ideal case of study to validate the proposal of this dissertation.

4.3.2. Relevant Technologies

In the previous section, GINSENG was defined as a project with strict requirements. As these requirements cannot be met if the deployment and the technologies are not carefully thought, this project has involved much planning before the first deployment.

First, there was the need to decide the operating system to use. As SICS is one of the partners of the project and Contiki was the OS that better fitted the general objectives of the project regarding real-time operation and the use of IP for compatibility, it was chosen as the operating system to implement the GINSENG technology.

Then, as the MAC layers that Contiki supported did not met the requirements of the project, it was decided to create a new MAC Layer based on a Time Division Multiple Access (TDMA) scheme and not to use Carrier Sense Multiple Access (CSMA) schemes as already implement in Contiki. This has one major cause: their operation is different and the reliability provided is also very distinct.

In fact, a CSMA MAC layer senses the medium each time it wants to transmit data, which causes collisions and delays in data transmission. On the opposite, in a TDMA MAC layer the available time is divided in slots and each sensor node has its specific slots to communicate thus avoiding collisions and most of possible delays.

Considering the previous paragraph, a MAC layer called GinMAC [15][16] was developed to fit the requirements of GINSENG. This MAC layer relies on a static schedule that is pre-programmed in the sensor nodes, using a tree topology to support multihop routing. In figure 21, it is possible to observe the example of slots definition for a given sensor node with 2 children and 1 parent.

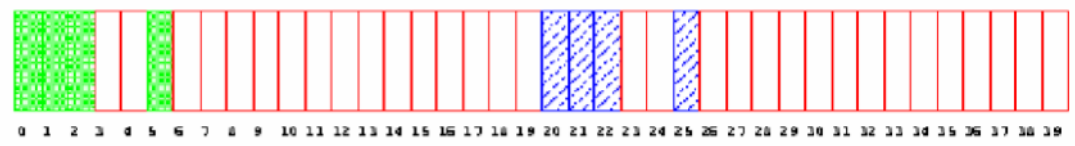

Figure 21 - GinMAC Slots Example

In this slot definition, the blank slots are slots when the node is sleeping, the green slots are the first slots to the normal transmission and the blue slots are the slots for the case when retransmission is needed. Basically, the first three slots of each group are upstream slots (one for the node itself, two for the children to send to the upper level) and the other is a downstream slot used for broadcast communication from the sink sensor node (the root node in the tree) to all the other nodes.

Considering this, it is easy to deduce that if the schedule is well adjusted the delay requirements are met, retransmissions can occur and most of the packets will arrive and even more important the energy consumption decreases because most of the time the sensor node is in sleep mode (radio turned off).

4.3.3. Industrial Scenario

Currently, GINSENG has one major industrial scenario that has been deployed physically in the Petrogal refinery of Sines. This scenario is not only representative from the industrial perspective of GINSENG project but also adjustable to the current epoch of GinMAC. This epoch has the duration of 1 second, allowing 100 slots of 10 milliseconds each (2.5 milliseconds pre-processing time, 5 milliseconds processing time and 2.5 milliseconds post-processing time). These slots are then divided through the nodes allowing up to 16 processing slots, 76 active (downstream and upstream) slots and 8 sleep slots. This leads us to the network topology, which in this case is a 3-2-1 tree with 15 nodes plus the sink node, as observed in figure 22.

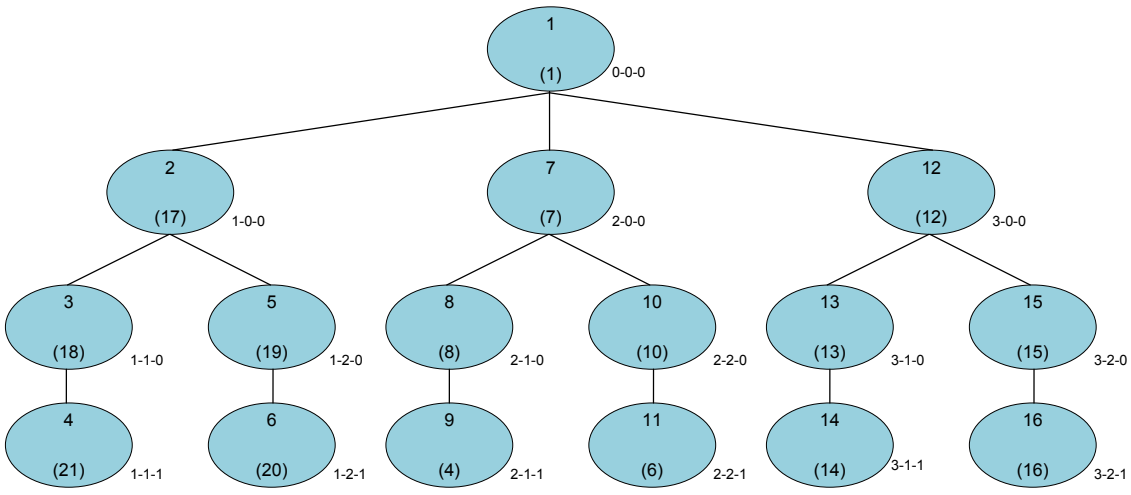

Figure 22 - GinMAC Tree

Also, in brackets it is included the mapping of real IDs to the tree topology, as in the real deployment the specific places to put the nodes have different numbering to adjust to the tree topology. These places are well defined, and may be located when looking at figures 23 and 24 that contain the definition and also the distances involved. One picture of the scenario is also included in Fig. 25 to exemplify the difficulties faced on the deployment.

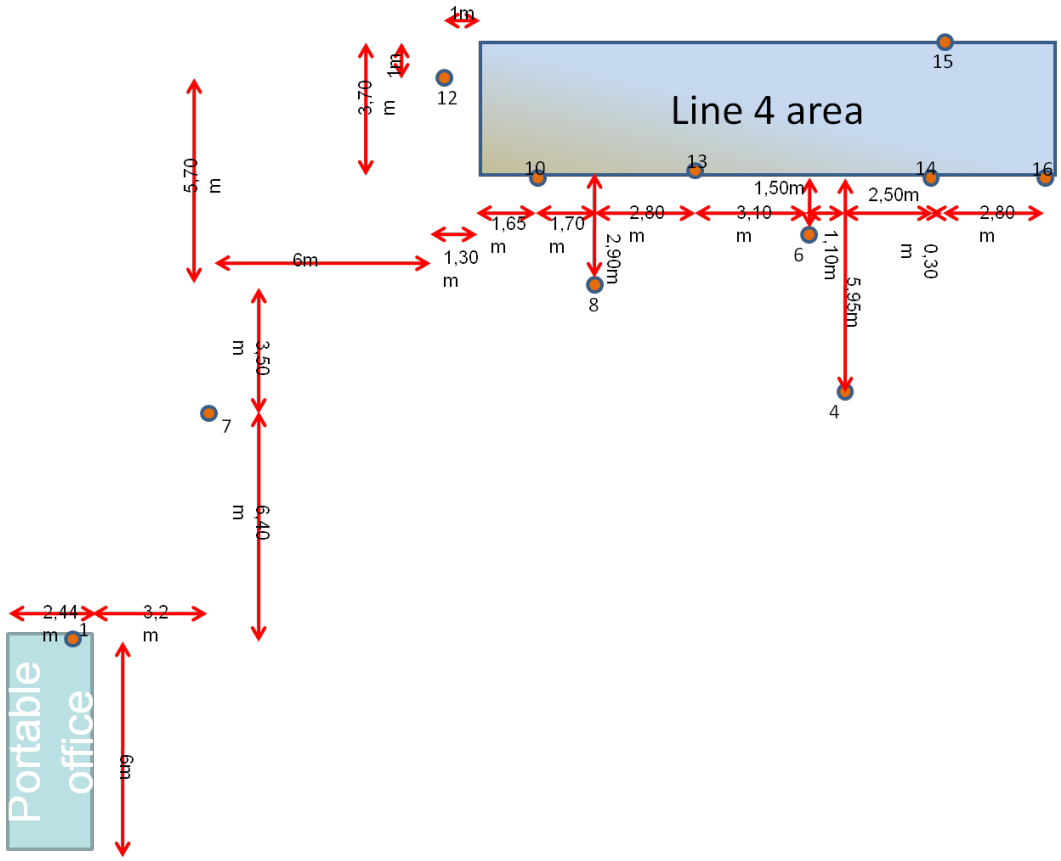

Figure 23 - Sines Line 4 Part 1

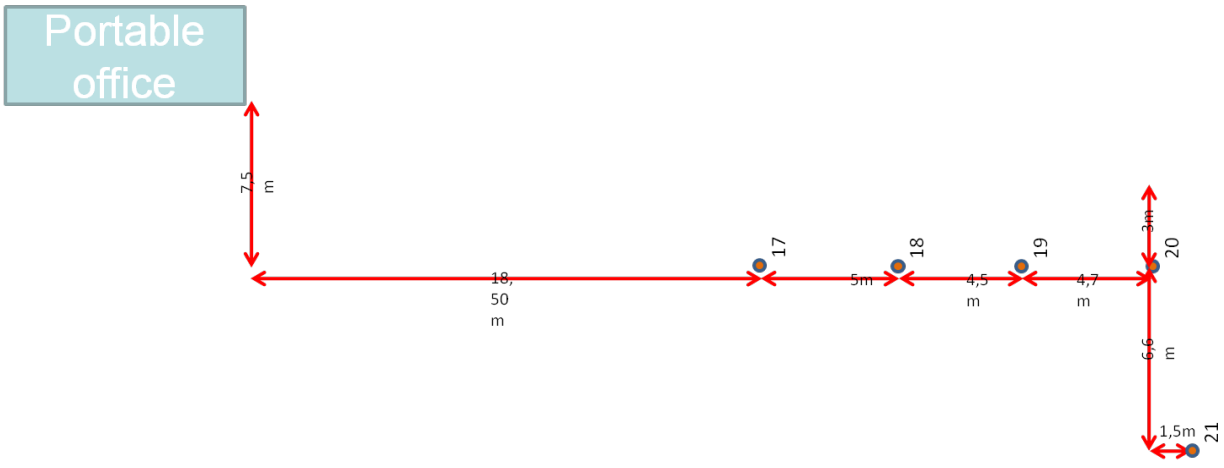

Figure 24 - Sines Line 4 Part 2

Figure 25 - GINSENG Industrial Scenario

Regarding the nodes themselves, they consist of Crossbow TelosB motes [17] which feature a MSP430 Microcontroller (MCU) working at 8 MHz, a CC2420 radio transceiver, 48 KB of ROM and 10KB of RAM. As operating system, currently they run Contiki, which provides full low-level support for GinMAC and also a multi layer (2 and 3) stack protocol called RIME [18] that is used to send messages across the network with 8 bits addressing (maximum of 256 nodes). Below GinMAC, the framing and radio transmission use the standard IEEE 802.15.4, the 2.4 GHz band using channel 16 and a bandwidth of 250 Kbps. GinMAC itself also provides layer 2 routing, which in GINSENG is done statically due to the referred topology.

Also important in GINSENG scenarios is the need for ATEX directives to be applied, which essentially state which devices and equipment can be used in environments with explosive atmospheres. In the next figure, it is possible to observe one of the containers needed to place motes at a refinery that creates additional effort on deployment of test equipment:

Figure 26 - ATEX Box

4.3.4. Security Requirements

Currently, the set of security requirements in GINSENG comprises authentication mechanisms to avoid intrusions, integrity mechanisms to avoid data corruption or adulteration and finally encryption mechanisms to assure confidentiality. Nevertheless, and despite the high priority of these requirements, the major factor to take in account is how the chosen mechanisms will affect the performance of the remaining modules and the overall system. In fact, as defined before the main requirements for GINSENG are related with performance assurances and therefore security requirements have lower priority.

4.4. Tests Results

The results for the defined tests, both on GINSENG project scenarios and integration scenarios, are presented and described below.

4.4.1. GINSENG Scenarios

In the following plots, the results in the GINSENG scenarios are presented.

The first results are related with memory consumption, namely ROM (Fig. 25) and RAM (Fig.26). The plots for both metrics show the consumption when the mote supports hardware cryptography (first bar) and when it does not (second bar). On each bar, it is possible to observe the base utilization (Contiki OS and GINSENG software) together with the sum of each module of the security model.

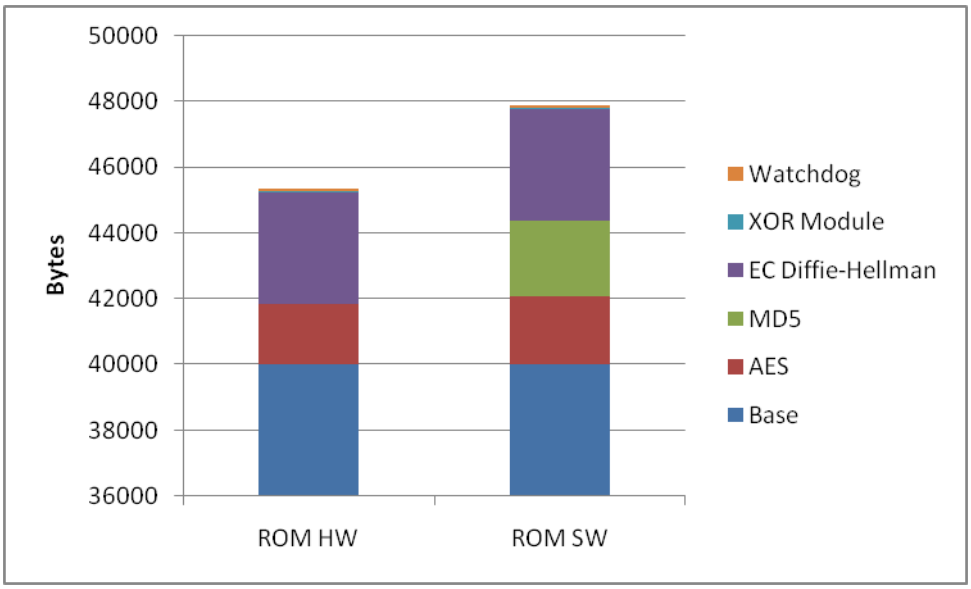

Figure 27 - Security ROM Usage

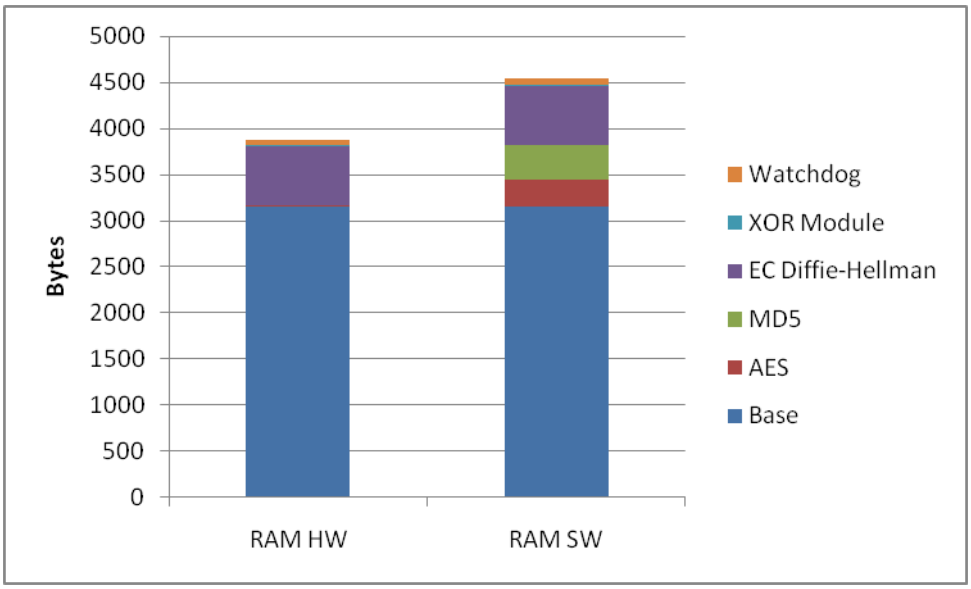

Figure 28 - Security RAM Usage

From these 2 plots, it is possible to conclude that the security model application is feasible given the limitations of the platform (48KB of ROM and 10KB of RAM). However, although RAM consumption is very low for both bars, when software cryptography is needed the amount of ROM for expansion is very small. This may pose a problem due to

the high base utilization of ROM for GINSENG, but as the chosen platform for this case is TelosB (with hardware cryptography support) there is enough space to insert more code.

Also, it is possible to validate the model requirement of RAM and ROM utilization. As the defined requirements were 2KB of RAM and 8KB of ROM at most, here we may conclude that even with all implementations at software level these resources never get past those values.

The second results demonstrate the end-to-end latency evaluation by tree levels. This is due to the tree topology of GINSENG, which normally affects the delay as multihop is needed. Level 1 is considered the tree branch below the root (sink node), level 2 the middle branch and level 3 the branch which includes the leaf nodes. For each of these levels, end-to-end latency (node to sink) was measured without security, with hardware based cryptography and also with software based cryptography. This evaluation can be observed in Figure 27.

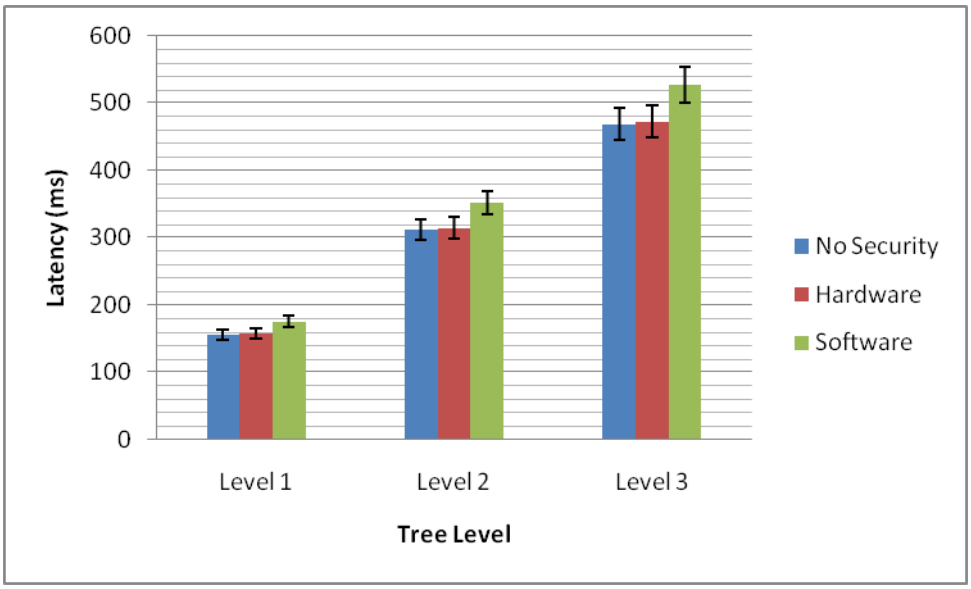

Figure 29 - Security Latency Impact

From this plot, we may conclude that latency is highly affected by the level at which the node is placed. In fact, the relation is roughly N x Level 1 Latency where N is the node's level. Additionally, the impact of the security model when hardware cryptography is used is very low and inside the margin of the standard deviation which produces a irrelevant difference. However, the same does not apply to software only based cryptography. In this case, the value is at most about 50 milliseconds above the non-secured environment. Even so, it is considered a good value as the requirements of the specific scenario state that everything below 1 second is inside the margin and the worst obtained value (~525 milliseconds) plus the standard deviation does not get past 560 milliseconds.

Finally, the last results represent the battery level (autonomy) of the network with an average from all the tree levels evaluated by the type of security applied. It was evaluated for the

network running with no security, with hardware based cryptography and with software only based cryptography. Also, for both hardware and software cryptography implementations, tests were conducted with and without the application of security levels through the monitoring done by the watchdog. The results are presented in the next figure:

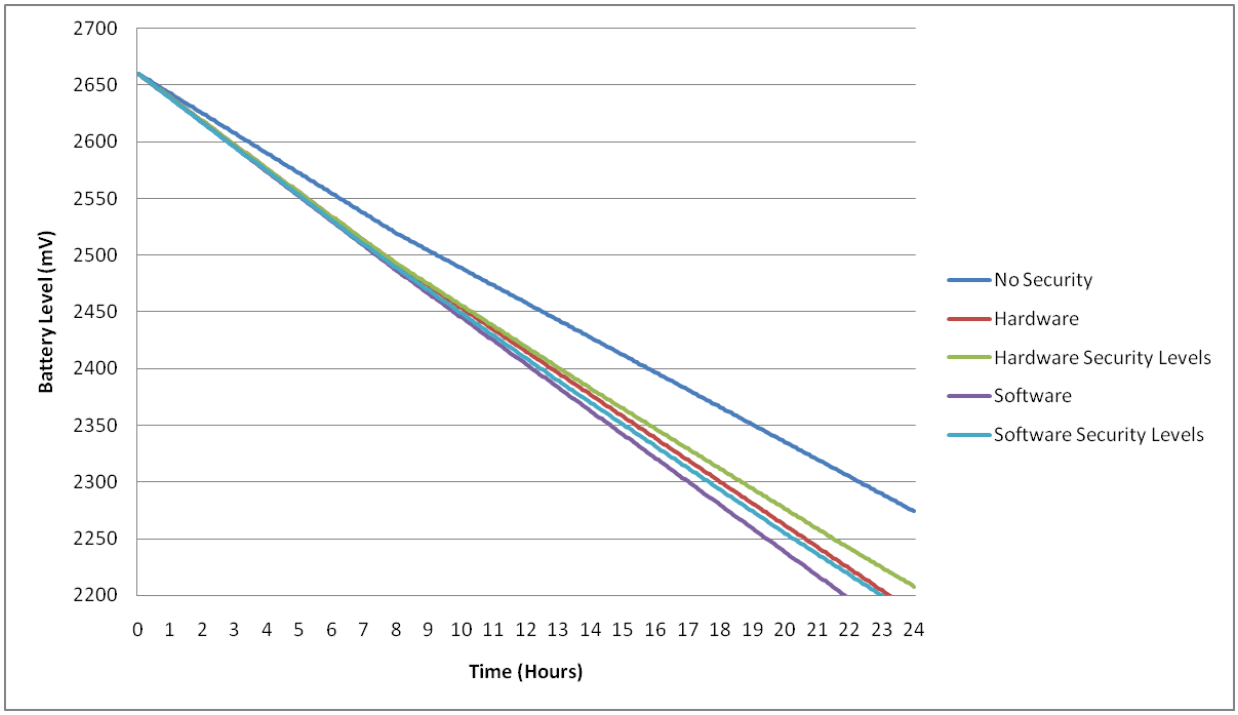

Figure 30 - Security Battery Discharge

From the lines at the above plot, the theory analyzed at sub-section 4.2.1 regarding the battery discharge rate being higher when the battery is full is proved. In fact, after 8 hours of running time the discharge rate began to stabilize at a lower value. Additionally, the decrease of 6 mV at each time is also observable as there are some small (at this scale) steps along the lines.

Other conclusion from this evaluation was that after the battery (two AA batteries) decreases below 2200 mV, the node stops working properly. This was later confirmed by the TelosB datasheet that states that below around 2100 mV the ADCs and other components stop working properly. Therefore, the results presented here only considered the interval until the value gets below 2200mV.

Regarding the results themselves, it is possible to observe that without security the autonomy of the nodes gets easily past the 24 hours (study duration). However, there is not much difference when comparing to hardware-based cryptography, especially when the security levels are dynamically adjusted to fit the remaining energy. As for software-based cryptography, the results show that the autonomy does not go after the 23 hours. However, this is still a very good value and shows a low impact (about 2 hours a day) for a security model with such a level of security relying solely on software implementations.

4.4.2. Contiki – TinyOS Integration

As explained on sub-section 4.1, integration tests between Contiki OS and TinyOS operating systems were conducted. Also, different architectures and different protocols implementation were integrated in order to ensure that the security model proposed in this thesis is in fact interoperable.

After setting up the scenario and turning on the motes, the first problem came to light. There was no communication because the duty cycle (radio ON/radio OFF) schemes of the MAC Layers in Contiki and TinyOS were different. The easiest solution was to use a NullMAC implementation (radio always ON), which led to communication occuring at layer 1. However, MAC Layers were still no communicating. This was a much harder to solve issue, and only after digging into hundreds of lines of code the problem became clear. There are differences in the implementation of IEEE 802.15.4 of both Contiki and TinyOS.

The most relevant one, and the only one that is important enough to mention, was the size of the MAC addresses. Contiki uses extended IEEE 802.15.4 addresses (64 bits) as TinyOS uses short IEEE 802.15.4 addresses. This is configurable and an option at the standard, which is defined at the MAC Frame (Fig. 29).

<table>
<tr><th>Octets:
2</th><th>1</th><th>0/2</th><th>0/2/8</th><th>0/2</th><th>0/2/8</th><th>0/5/6/10/
14</th><th>variable</th><th>2</th></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

Figure 31 - 802.15.4 MAC Frame

In this figure, it is possible to observe that the address fields can be 0, 2 (short) or 8 bytes (extended). Also, there is a parameter to configure which addresses are being used inside the Frame Control field (Fig. 30).

Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	Ack. Request	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

Figure 32 – Frame Control Field

Inside this field, the 10 and 11 bits define which type of addresses is being used at the current frame depending if they are set to 0 or to 1. This can be observed in the next figure (Fig. 31).

Addressing mode value b_1 b_0	Description
00	PAN identifier and address fields are not present.
01	Reserved.
10	Address field contains a 16-bit short address.
11	Address field contains a 64-bit extended address.

Figure 33 - Destination Addressing Mode

After understanding this issue, the solution seemed to be easy. Change one of the operating systems' configurations to use the same type address of the other. However, the solution was not so evident because upper layers also depend on these addresses. In fact, 6LoWPAN uses the 802.15.4 address to derivate the suffix for the IPv6 address and the algorithm is once again different in both TinyOS (BLIP) and Contiki (SICSLoWPAN). As the author of this thesis is more familiar with the implementation in Contiki, it was decided to change the address size in Contiki and adjust the algorithm at SICSLoWPAN that derivates the suffix.

Next, the communication stacks were finally communicating but packets where not arriving at the application layer. It was one more difference of the 6LoWPAN implementation, now regarding the header compression algorithms. After adjusting both implementations to use the most basic header compression algorithm (different types are available), data started arriving at the application of both motes without problems. This ran successfully for 24 hours with the security model on at the highest level of security with both hardware and software based cryptography.

Chapter 5
Dissertation Work Plan

The work related with this dissertation followed a simple methodology in order to keep a good workflow without too much overhead. During the first semester, the work plan for the whole year was defined using Gantt charts (Fig. 34 and 35) containing the major tasks and the time period where they should be developed and finished. After each task, validation tests occurred and, if needed, small adjustments were made.

Additionally, weekly meetings with the thesis advisors were also a part of the methodology, allowing discussion of completed tasks and feedback to solve problems and optimize developed software.

Finally, the writing of scientific publications was also the focus of this work. In fact, two papers were submitted to international ISI journals to improve the contribution in terms of scientific relevance. These had also suggestions and reviews from the advisors, which were decisive to accomplish good level papers.

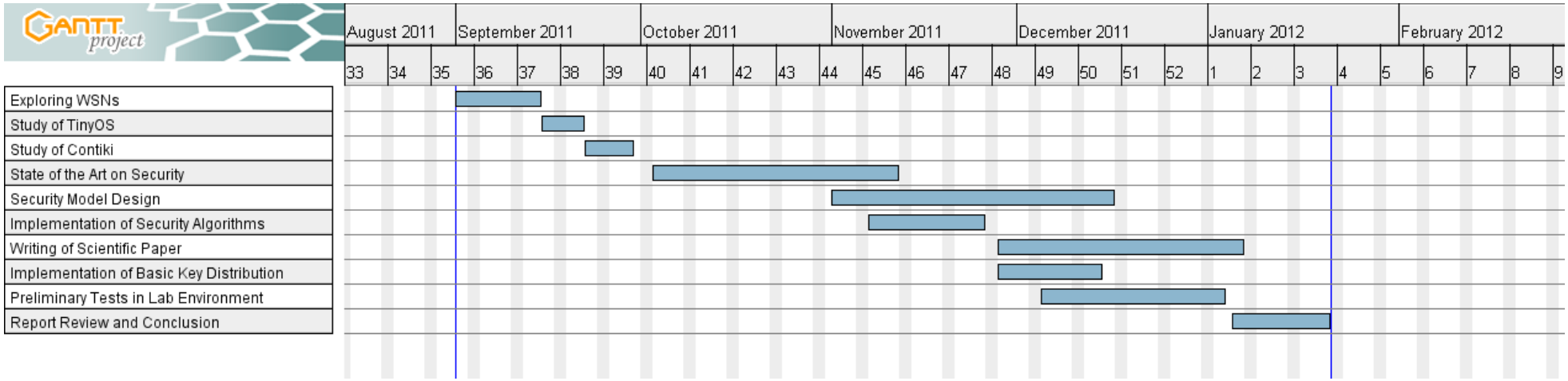

Figure 34 – 1st Semester Gantt Chart

During the first semester, the work was mostly focused on the study of the technologies and state of the art on security. However, to achieve a proposal for the security model, requirements were defined and tests were also conducted in order to evaluate which algorithms could fit those requirements while fitting the constraints of the architectures of WSNs. These testes provided some conclusions and also were part of a scientific paper that was submitted to an international ISI journal.

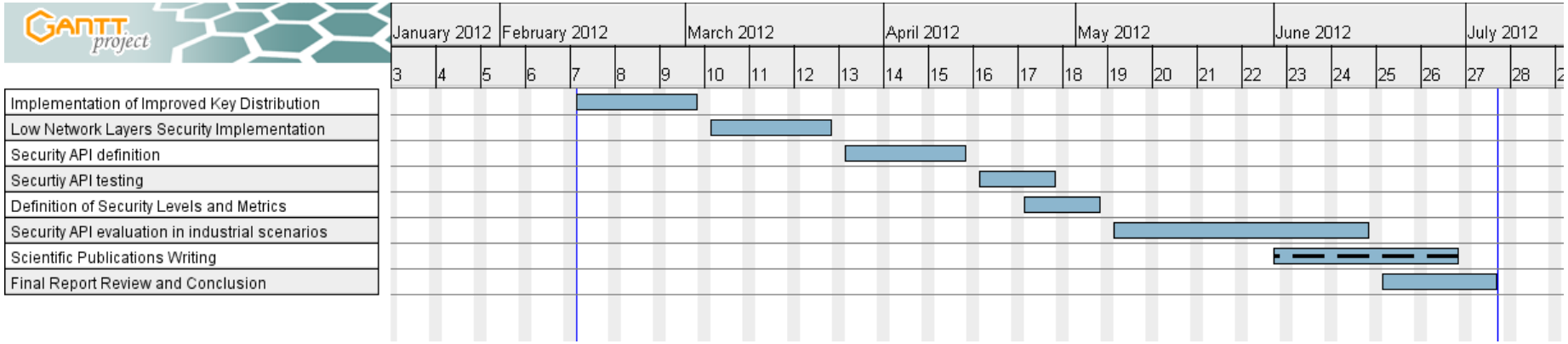

Figure 35 - 2nd Semester Gantt Chart

During the second semester, the work was more focused on the conclusion of the model specification and its implementation for several hardware platforms. While testing the model to evaluate its performance, compatibility, interoperability and portability, several other minor tests occurred to define security levels and metrics to trigger pre-determined actions. Also important was the API definition and documentation, which enables any user to use the security model on any platform without any predicted issues.

Finally, the model was fully validated with the tests due to the conclusions they allowed the author to gather. These results both with the security model definition and innovations allowed the author to write one more paper to be submitted to a high level international ISI journal together with this final report that covers the whole dissertation work and conclusions during the two semesters.

Chapter 6
Conclusions and Future Work

During the 2 semesters that this dissertation's work has been carried, much was learned and new challenges appeared at each step, always involving hard work to be overcome. Although these difficulties appeared and created extra effort, the global auto evaluation of the results is very positive, as the main goals defined at the intermediate report were fulfilled in their most relevant essence.

In fact, a fully functional security model with security, performance, interoperability and reliability was obtained. This is supported by the tests included in chapter 4, which clearly prove that the proposed security model is valid and easily integrated. This last statement is in fact well proven, as the integration issues to test compatibility, portability and interoperability were all related with protocols and underlying operating systems not following the standards or using different options of implementation.

Also relevant was the submission of two scientific papers for international ISI journals. These are included in annex and are the following:

- ✓ "Wireles Sensor Networks Security in Industrial Environments", submitted to IEEE Transactions on Industrial Electronics.

- ✓ "WSNOpenSec - An Interoperable Security Model for Wireless Sensor Networks", submitted to Elsevier Pervasive and Mobile Computing.

Although the model proposed and validated in this work is an improvement over other solutions and due to its open source characteristic may be attractive to the open source community represented by Contiki and TinyOS, there is still much room for improvements. These improvements can be summarized by the following topics:

- ✓ Improve attack detection and dynamic security measures.

- ✓ Make adjustments or find new algorithms to enhance the performance of software cryptographic implementations.

- ✓ Optimize key distribution and key renewal procedures.

- ✓ Study and evaluate better layer 1 security mechanisms with equivalent performance.

- ✓ Study and test authentication mechanisms to join the network for the first time and to improve general authentication. Radio patterns, which are like fingerprints of the radio transceivers, could be an option.

Additionally, more improvements are always possible and technology is evolving at a rhythm that hungrily demands new security systems and mechanisms as privacy and security concerns arise. Hence, the subject of this thesis and the results obtained here can be used as a point of start for future work of other students and researchers.

Finally, there are also contacts with the open source community responsible for Contiki in order to publish the work of the proposed security model as part of the operating system. It is predicted that this possibility comes true, as Contiki does not have any relevant security

mechanisms embedded on it (unlinke TinyOS that already has some security, mainly IEEE 802.15.4 security).

References

[1] M. Welsh, "7th European Conference on Wiress Sensor Networks (EWSN 2010) Keynote", Coimbra, Feb. 2010. [Online]. Available: http://blip.tv/matt-welsh/matt-welsh-ewsn-2010-keynote-3272727

[2] University of California. (2011, Jan.) TinyOS. [Online]. Available: http://www.tinyos.net

[3] Swedish Institute of Computer Science. (2011, Jan.) Contiki. [Online]. Available: http://www.contiki-os.org

[4] Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," Communications Magazine, IEEE, vol. 40, no. 8, pp. 102–114, Aug 2002.

[5] Mo Sha. (2011, Jan.) TelosB Mote. [Online]. Available: http://students.cec.wustl.edu/~ms31/figs/telosb.png

[6] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LRWPANs), IEEE Std. 802.15.4, Sep. 2006. [Online]. Available: http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf

[7] ZigBee Alliance. (2011, Jan.) ZigBee. [Online]. Available: http://www.zigbee.org

[8] Z. Shelby and C. Bormann, 6LoWPAN: The Wireless Embedded Internet, 1st ed. Hoboken, NJ: Wiley, Jan. 2010.

[9] D. Chen and P. K. Varshney, "QoS support in wireless sensor networks: a survey," in Proc. International Conference on Wireless Networks (ICWN '04), pp. 227-233, Las Vegas, Nev, USA, June 2004.

[10] M. Younis, K. Akkaya, M. Eltoweissy, and A. Wadaa, "On handling QoS traffic in wireless sensor networks," System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on, 2004, p. 10 pp.

[11] Dunkels, A.; Gronvall, B.; Voigt, T.; , "Contiki - a lightweight and flexible operating system for tiny networked sensors," *Local Computer Networks, 2004. 29th Annual IEEE International Conference on* , vol., no., pp. 455- 462, 16-18 Nov. 2004

[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. 2000. System architecture directions for networked sensors. In *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems* (ASPLOS-IX). ACM, New York, NY, USA, 93-104.

[13] University of California. (2011, Jan.) nesC: A Programming Language for Deeply Networked Systems . [Online]. Available: http://nescc.sourceforge.net

[14] C. J. Sreenan. (2008, Sep.) Ginseng - Performance Control in WSNs. [Online]. Available: ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/necs/fp7-fact-sheet-ginseng en.pdf

[15] O'Donovan T, Brown J, Roedig U, Sreenan CJ, do Ó J, Dunkels A, Klein A, Silva JS, Vassiliou V, Wolf L. GINSENG - Performance control in Wireless Sensor Networks. Proceeding of the 7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010; 1–3.

[16] Schmitt JB, Roedig U. Worst case dimensioning of Wireless Sensor Networks under uncertain topologies. Proceedings of the 3rd IEEE International Symposium on Modeling

and Optimization in Mobile, Ad Hoc, and Wireless Networks, (WiOpt'05) Workshop on Resource Allocation in Wireless Networks, Riva del Garda, Italy, 2005.

[17] C. Technology®. (2011, Jan.) TelosB™ Datasheet. [Online]. Available: http://www.willow.co.uk/TelosB Datasheet.pdf

[18] A. Dunkels. Rime - a lightweight layered communication stack for sensor networks. In Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session, Delft, The Netherlands, January 2007.

[19] D. W. Carman, P. S. Krus, and B. J. Matt. Constraints and approaches for distributed sensor network security. Technical Report 00-010, NAI Labs, Network Associates, Inc., Glenwood, MD, 2000.

[20] S. Ganeriwal, S. Capkun, C.-C. Han, and M. B. Srivastava. Secure time synchronization service for sensor networks. In WiSe '05: Proceedings of the 4th ACM workshop on Wireless security, pages 97–106, New York, NY, USA, 2005. ACM Press.

[21] S. Capkun and J.-P. Hubaux. Secure positioning in wireless networks. IEEE Journal on Selected Areas in Communications, 24(2):221–232, 2006.

[22] L. Lazos and R. Poovendran. Serloc: Robust localization for wireless sensor networks. ACM Trans. Sen. Netw., 1(1):73–100, 2005.

[23] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. Computer, 35(10):54–62, 2002.

[24] J. Douceur. The sybil attack. In Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), February 2002.

[25] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In Proceedings of the third international symposium on Information processing in sensor networks, pages 259–268. ACM Press, 2004.

[26] X. Wang, W. Gu, K. Schosek, S. Chellappan, and D. Xuan. Sensor network configuration under physical attacks. Technical Report Technical Report (OSU-CISRC-7/ 04-TR45), Dept. of Computer Science and Engineering, The Ohio-State University, July 2004.

[27]G. Gaubatz, J.P. Kaps, and B. Sunar. Public key cryptography in sensor networks - revisited. In 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004), 2004.

[28] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In Proceedings of the 9th ACM conference on Computer and communications security, pages 41–47. ACM Press, 2002.

[29] J. Deng, R. Han, and S. Mishra. INSENS: intrusion-tolerant routing in wireless sensor networks. In Technical Report CU-CS-939-02, Department of Computer Science, University of Colorado, 2002.

[30] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In Proceedings of IEEE Symposium on Security and Privacy, May 2005

[31] J. Deng, R. Han, and S. Mishra. Countermeasuers against traffic analysis in wireless sensor networks. Technical Report CU-CS-987-04, University of Colorado at Boulder, 2004.

[32] X. Wang, W. Gu, S. Chellappan, Dong Xuan, and Ten H. Laii. Search-based physical attacks in sensor networks: Modeling and defense. Technical report, Dept. of Computer Science and Engineering, The Ohio-State University, February 2005.

[33] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In The Second USENIX Workshop on Electronic Commerce Proceedings, Oakland, California, 1996.

[34] Sastry, N. and Wagner, D. Security Considerations for IEEE 802.15.4 Networks. In WiSe '04: Proceedings of the 3rd ACM workshop on Wireless security, pages 32–42, New York, NY, USA, 2004. ACM Press.

[35] D. Gascón. (2011, Jan.) IEEE 802.15.4 and ZigBee Security. [Online]. Available: http://www.sensor-networks.org/index.php?page=0903503549

[36] J. Cobb, E. Rotvold, J. Potter. (2010) 'WirelessHART Security Overview'. [Online]. Available: http://www.hartcomm.org/protocol/training/resources/wiHART_resources/Security_Overview_LIT114.pdf

[37] Daemen J, Rijmen V. The Design of Rijndael: AES - The Advanced Encryption Standard. First edn., Springer, 2002.

[38] National Institute of Standards and Technology. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197 Nov 2001. URL http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[39] Law YW, Doumen J, Hartel P. Survey and benchmark of block ciphers for Wireless Sensor Networks. ACM Transactions on Sensor Networks TOSN Feb 2006; 2(1):65–93.

[40] Karlof C, Sastry N, Wagner D. TinySec: A link layer security architecture for Wireless Sensor Networks. Proceeding of the 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, MD, 2004; 162–175.

[41] Rivest R. The MD5 Message-Digest Algorithm. RFC 1321 Apr 1992. URL http://tools.ietf.org/html/rfc1321.

[42] National Institute of Standards and Technology. Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180-2 Aug 2002. URL http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

[43] Granjal J, Silva R, Monteiro E, Silva JS, Boavida F. Why is IPSec a viable option for Wireless Sensor Networks. Proceedings of the Fifth IEEE International Conference on Mobile Ad-hoc and Sensor Systems, 2008; 802–807.

[44] Chipcon. CC2420 Datasheet Jun 2004. [Online]. Available: http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf.

[45] Didla S., Ault A, Bagchi S. Optimizing AES for embedded devices and Wireless Sensor Networks. Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities, 4, 2008; 10.

[46] Gladman, B. "Optimized AES Implementation". Fev. 2012 [Online]. Available: http://www.gladman.me.uk/

[47] Diffie, W.; , "The first ten years of public-key cryptography," *Proceedings of the IEEE* , vol.76, no.5, pp.560-577, May 1988

[48] Ripley. (2011, Jan.) Diffie Hellman Key Exchange. [Online]. Available: http://www.ripley9.net/wp-content/uploads/2011/12/DiffieHellmanKeyExchange2.png

[49] Chris, K., Cockrum: "Implementation of an Elliptic Curve Cryptosystem on an 8-bit Microcontroller", Springer, Heidelberg (2009)

[50] Wang Qingxian; , "The application of elliptic curves cryptography in embedded systems," *Embedded Software and Systems, 2005. Second International Conference on* , vol., no., pp. 4 pp., 16-18 Dec. 2005

[51] Chung, Antony and Roedig, Utz (2007) *Efficient Key Establishment for Wireless Sensor Networks Using Elliptic Curve Diffie-Hellman.* In: Proceedings of the 2nd European Conference on Smart Sensing and Context (EUROSSC2007), October 2007, Kendal, UK.

[52] Wikimedia. "Simple Elliptic Curve". Apr. 2012 [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/d/da/Elliptic_curve_simple.svg

[53] Koblitz, N. (1987). "Elliptic curve cryptosystems". Mathematics of Computation 48 (177): 203–209. JSTOR 2007884.

[54] Miller, V. (1985). "Use of elliptic curves in cryptography". CRYPTO 85: 417–426. DOI:10.1007/3-540-39799-X_31.

[55] NSA. "The Case for Elliptic Curve Cryptography". Apr. 2012 [Online]. Available: http://www.nsa.gov/business/programs/elliptic_curve.shtml

[56] NXP Semiconductors. (2011, Jan.) Jennic JN5139. [Online]. Available: http://www.jennic.com/products/wireless_microcontrollers/jn5139

[57] M. Kramer and A. Gerldy, "Energy Measurements for MicaZ Node," Fachgespräch "Dahtlose Sensornetze", GI/ITG KuVS, 2006

Annex A – API

AES-128 Hardware

```
/*********************************************************************
 * Store a given key in CC2420's RAM (max. 2 simultaneous keys).
 *
 * Parameters:
 *        key             ->      the 128 bits (16 bytes) key to set
 *        index   ->      the key index (0 or 1)
 *
 */
void cc2420_inline_aes_set_key(uint8_t *key, int index);

/*********************************************************************
 * Generate a MAC for a given sequence of bytes using a certain key
 * and one of the folowing modes:
 *
 * -> AES-CBC-MAC-32:       32 bits (4 bytes) MAC
 * -> AES-CBC-MAC-64:       64 bits (8 bytes) MAC
 * -> AES-CBC-MAC-128:      128 bits (16 bytes) MAC
 *
 * Parameters:
 *        data            ->      the input sequence of bytes
 *        len                     ->      the length of data
 *        mac                     ->      a byte array to store the generated MAC
 *        maclen          ->      the length of the desired MAC (4, 8 or 16)
 *        key_index       ->      the index of the key to use (0 or 1)
 *
 */
void cc2420_inline_aes-cbc-mac_compute(uint8_t *data, int len, uint8_t *mac, int maclen, int
key_index);

/*********************************************************************
 * Verify a given sequence of bytes using a given MAC, a certain key
 * and one of the folowing modes:
 *
 * -> AES-CBC-MAC-32:       32 bits (4 bytes) MAC
 * -> AES-CBC-MAC-64:       64 bits (8 bytes) MAC
 * -> AES-CBC-MAC-128:      128 bits (16 bytes) MAC
 *
 * Parameters:
 *        data            ->      the input sequence of bytes
 *        len                     ->      the length of data
 *        mac                     ->      the MAC to verify the data
 *        maclen          ->      the length of the MAC (4, 8 or 16)
 *        key_index       ->      the index of the key to use (0 or 1)
 *
 * Return:
 *        valid           ->      1 if MAC is valid and 0 if MAC is not valid
 *
 */
int cc2420_inline_aes-cbc-mac_verify(uint8_t *data, int len, uint8_t *mac, int maclen, int
key_index);

/*********************************************************************
 * Encrypt a given sequence of bytes using a certain key and the
 * AES-CTR mode
```

```
 *
 * Parameters:
 *      data            ->      the input sequence of bytes
 *      len                     ->      the length of data
 *      key_index       ->      the index of the key to use (0 or 1)
 *
 * Observations:
 *      Encrypted data overrides the sequence of bytes inside the
 *      variable data.
 *
 */
void cc2420_inline_aes-ctr_encrypt(uint8_t *data, int len, int key_index);

/*---------------------------------------------------------------------*
 * Decrypt a given sequence of bytes using a certain key and the
 * AES-CTR mode.
 *
 * Parameters:
 *      data            ->      the input sequence of bytes
 *      len                     ->      the length of data
 *      key_index       ->      the index of the key to use (0 or 1)
 *
 * Observations:
 *      Decrypted data overrides the sequence of bytes inside the
 *      variable data.
 *
 */
void cc2420_inline_aes-ctr_decrypt(uint8_t *data, int len, int key_index);

/*---------------------------------------------------------------------*
 * Generate a MAC for a given sequence of bytes and encrypt the
 * whole result using a certain key and one of the folowing modes:
 *
 * -> AES-CCM-32:       32 bits (4 bytes) MAC
 * -> AES-CCM-64:       64 bits (8 bytes) MAC
 * -> AES-CCM-128:      128 bits (16 bytes) MAC
 *
 * Parameters:
 *      data            ->      the input sequence of bytes
 *      len                     ->      the length of data
 *      maclen          ->      the length of the desired MAC (4, 8 or 16)
 *      key_index       ->      the index of the key to use (0 or 1)
 *
 * Observations:
 *      Encrypted data overrides the sequence of bytes inside the
 *      variable data.
 *
 * WARNING:
 *      Data array must have a certain free number of bytes in the end
 *      according to the desired MAC length.
 *
 */
void cc2420_inline_aes-ccm_encode(uint8_t *data, int len, int maclen, int key_index);

/*---------------------------------------------------------------------*
 * Decrypt a given sequence of bytes and verify the included MAC
 * using a certain key and one of the folowing modes:
 *
 * -> AES-CCM-32:       32 bits (4 bytes) MAC
 * -> AES-CCM-64:       64 bits (8 bytes) MAC
 * -> AES-CCM-128:      128 bits (16 bytes) MAC
```

```
 *
 * Parameters:
 *      data            ->      the input sequence of bytes
 *      len                     ->      the length of data
 *      maclen          ->      the length of the included MAC (4, 8 or 16)
 *      key_index       ->      the index of the key to use (0 or 1)
 *
 * Return:
 *      valid           ->      1 if MAC is valid and 0 if MAC is not valid
 *
 * Observations:
 *      Decrypted data overrides the sequence of bytes inside the
 *      variable data.
 *
 */
int cc2420_inline_aes-ccm_decode(uint8_t *data, int len, uint8_t *mac, int maclen, int key_index);

/*****************************************************************
 * Write a given 802.15.4 frame to CC2420's RX FIFO.
 *
 * Parameters:
 *      frame           ->      the input frame
 *      framelen        ->      the length of the frame
 *
 * Format:
 *      LENGTH (1 byte) | FRAME (framelen bytes) | CRC-16 (2 bytes)
 *
 * Observations:
 *      LENGTH includes the FRAME length, the MAC size (if applicable)
 *      and the CRC-16 length.
 *
 */
void cc2420_write_rxfifo(const void *frame, unsigned short framelen);

/*****************************************************************
 * Write a given 802.15.4 frame to CC2420's TX FIFO.
 *
 * Parameters:
 *      frame           ->      the input frame
 *      framelen        ->      the length of the frame
 *
 * Format:
 *      LENGTH (1 byte) | FRAME (framelen bytes) | CRC-16 (2 bytes)
 *
 * Observations:
 *      LENGTH includes the FRAME length, the MAC size (if applicable)
 *      and the CRC-16 length.
 *
 */
void cc2420_write_txfifo(const void *frame, unsigned short framelen);
```

AES-128 Software

```
/*****************************************************************
 * Encrypt a given sequence of bytes using a certain key and the
 * AES-128 software module
 *
 * Parameters:
 *      data            ->      the input sequence of bytes
```

```
 *      len                     ->      the length of data
 *      key                     ->      the key to use
 *
 * Observations:
 *      Encrypted data overrides the sequence of bytes inside the
 *      variable data.
 *
 */
void software_aes_encrypt(uint8_t *data, int len, uint8_t *key);

/*****************************************************************
 * Decrypt a given sequence of bytes using a certain key and the
 * AES-128 software module
 *
 * Parameters:
 *      data                    ->      the input sequence of bytes
 *      len                     ->      the length of data
 *      key                     ->      the key to use
 *
 * Observations:
 *      Decrypted data overrides the sequence of bytes inside the
 *      variable data.
 *
 */
void software_aes_decrypt(uint8_t *data, int len, uint8_t *key);
```

MD5

```
/*****************************************************************
 * Obtain the MD5 hash of a given sequence of bytes
 *
 * Parameters:
 *      data                    ->      the input sequence of bytes
 *      len                     ->      the length of data
 *      hash                    ->      the variable where to place the hash
 *
 * Observations:
 *      Hash has to be an array with 16 bytes.
 *
 */
void md5_hash(uint8_t *data, int len, uint8_t *hash);
```

Elliptic Curve Diffie-Hellman

```
/*****************************************************************
 * Set the elliptic curve to use by EC-DH
 * y2 = x3 + ax + b
 *
 * Parameters:
 *      a                       ->      a in the equation
 *      b                       ->      b in the equation
```

```
 *      r                       ->      underlying field of the curve
 *
 * Observations:
 *      None.
 *
 */
void set_ecdh_curve(int a, int b, int r);

/*****************************************************************
 * Set the elliptic curve base point to use by EC-DH
 *
 * Parameters:
 *      x                       ->      x coordinate
 *      y                       ->      y coordinate
 *
 * Observations:
 *      None.
 *
 */
void set_ecdh_base_point(int x, int y);


/*****************************************************************
 * Get EC-DH private key
 *
 * Parameters:
 *      key                     ->      the variable where to put the private key
 *
 * Observations:
 *      Key has to be an array with 32 bytes.
 *
 */
void get_ecdh_private(uint8_t *key);
```

```
/*****************************************************************
 * Get EC-DH public key
 *
 * Parameters:
 *      key                     ->      the variable where to put the public key
 *
 * Observations:
 *      Key has to be an array with 32 bytes.
 *
 */
void get_ecdh_public(uint8_t *key);


/*****************************************************************
 * Get EC-DH derived symmetric key
 *
 * Parameters:
 *      key                     ->      the variable where to put the symmetric key
 *
 * Observations:
 *      Key has to be an array with 16 bytes.
 *
 */
void get_ecdh_symmetric(uint8_t *key);
```

XOR Module

```
/*****************************************************************
 * Obtain the XOR of a byte array with counters module and overflow
 * prevention.
 *
 * Parameters:
 *      data            ->      the input sequence of bytes
 *      len                     ->      the length of data
 *      N                       ->      the N (counter) to use
 *
 * Observations:
 * XORed data overrides the sequence of bytes inside the
 * variable data.
 *
 */
void get_xor(uint8_t *data, int len, int N);
```

Watchdog

```
/*********************************************************************
 * Renew Key
 *
 * Parameters:
 *      index           ->      0 if layer 2 key, 1 layer 3
 *
 * Observations:
 * None.
 *
 */
void renew_key(int index);


/*********************************************************************
 * Change Security Level
 *
 * Parameters:
 *      level           ->      0 Extreme
 *                              ->      1 High
 *                              ->      2 Medium
 *                              ->      3 Low
 *                              ->      4 Very Low
 *
 * Observations:
 * None.
 *
 */
void set_security_level(int index);

/*********************************************************************
 * Obtain Security Level
 *
 * Return:
 *      level           ->      0 Extreme
 *                              ->      1 High
 *                              ->      2 Medium
 *                              ->      3 Low
 *                              ->      4 Very Low
 *
 * Observations:
 * None.
 *
 */
int get_security_level();

/*********************************************************************
 * Set custom security level
 *
 * Parameters:
 *      l1              ->      XOR (1 if active, 0 othwrwise)
 *      l2              ->      802.15.4 Security (0 CCM, 1 CTR, 2 CBC)
 *      l3              ->      Protocol Security (0 Auth & Confidentiality)
 *                      ->      Protocol Security (1 Confidentiality)
 *                      ->      Protocol Security (2 Auth)
 * Observations:
 * None.
 *
 */
void set_custom_level(int l1, int l2, int l3);
```

Annex B – Wireles Sensor Networks Security in Industrial Environments

Wireless Sensor Networks Security in Industrial Environments

André Gomes, Jorge Granjal, Edmundo Monteiro, Jorge Sá Silva
Department of Informatics Engineering
University of Coimbra
Pólo 2 - Pinhal de Marrocos, 3030–290 Coimbra, PORTUGAL
{asng, jgranjal, edmundo, sasilva}@dei.uc.pt

***Abstract*—In today's world, Wireless Sensor Networks field of study is widely discussed among the research community, being expected that this trend will continue further. However, and even though some valid proposals already exist, security mechanisms to apply to these networks are still in a very embryonic state. Together with this lack of stable and proven mechanisms, there is also a void when it comes to apply security in industrial environments with real-time requirements. This article aims to evaluate security algorithms and verify which ones could be applied to such environments, mainly when there is a strict window to send/receive data. Also, hardware and software implementations are studied and compared using an experimental setup based on TelosB motes, being characterized by message losses, time/latency and energy consumption.**

***Index Terms*—Wireless sensor networks, security, real-time, industrial environments.**

I. Introduction

When it comes to research work regarding performance control in Wireless Sensor Networks (WSNs) that can be applied to real-time environments, ICT FP7 GINSENG – Performance Control in Wireless Sensor Networks [1] is one of the well-known current European projects. This project, in a brief summary, aims to create WSNs that meet demanding performance specifications and that can be applied in industrial environments where real-time operation is critical. Although most modules are almost completely developed and working properly, security is an aspect that has not been taken in account yet. This has one major cause – the difficulty of finding compatible mechanisms that do not affect performance.

Currently, the set of requirements comprises authentication mechanisms to avoid intrusions, integrity mechanisms to avoid data corruption or adulteration and finally encryption mechanisms to assure confidentiality. Once again, and despite the high priority of these requirements, the major factor to take in account is how the chosen mechanisms will affect the performance of the remaining modules and the overall system. Bearing this in mind, and as a first step, this article evaluates some well-known algorithms that can fulfill such requirements and how they affect the current system, which does not have any security level. Moreover, it also takes in account that the current GINSENG system uses a Time Division Multiple Access (TDMA) MAC Layer called GinMAC [2][3], and each node in the network has a strict slot of time in which it can communicate with the others.

Finally, preliminary conclusions are taken and future work with their basis is defined, contributing to a future security model that may be applied to any TDMA MAC Layer and even different platforms or operating systems.

II. Related Work

During many years, RC5 proposed by Rivest, R. and Skypjack introduced by NIST have been considered by the research community as the most relevant and suitable security algorithms for WSNs when it comes to symmetric block ciphers [4]. However, more recently technology has evolved and a new algorithm has taken its place as the most adopted and secure approach without compromising the low resource requirements of WSNs - Advanced Encryption Standard (AES) [5]. AES was initially published by Daemen, J. and Rijmen, V. as Rijndael in 1998, but with the selection process required for standardization it only came out as AES in 2001 [6]. Later, researchers have developed implementations for WSNs with optimizations and today it is considered the most preferable option [7]. Therefore, when it comes to evaluate possible security algorithms for specific scenarios AES has to be considered as the best candidate for data protection with symmetric block ciphers, but others should not be taken apart from any comparison study. However, as AES took the main role for data protection, most of the current days embedded radios for WSNs already have its cipher implemented using hardware which leads us to great results that cannot be matched by software implementations of other algorithms in terms of speed and resources utilization.

Also, it is important to evaluate algorithms that are capable of providing integrity, non-repudiation and authentication without requiring powerful computational resources to run. In this area, industry already uses algorithms such as Message-Digest algorithm 5 (MD5) [8] and Secure Hash Algorithm 1 (SHA-1) [9] that can meet these requirements.

Regarding security algorithms comparison, there are already some proposals that try to evaluate well-known and proven implementations to general scenarios without specific requirements. However, these comparisons usually do not take in account that some platforms already have hardware based security modules and that even when a security algorithm seems to fit the platform resources it may not fit a specific application at all because of its specific requirements. For

instance, Granjal, J. et al [10] evaluated algorithms such as Secure Hash Algorithm 1 (SHA1), Secure Hash Algorithm 2 (SHA2), AES and Triple Data Encryption Standard (3DES) but did not try different key sizes neither some available hardware implementations of AES available in radio transceivers such as the Chipcon CC2420 [11]. Others, like Didla, S. et al [12], extended their study past the software and also compared the hardware approach but didn't use its full potential as they let the decryption process out of the study. In this work, the main objective is to fill these gaps and provide a more comprehensive analysis of what can be used to provide security without affecting performance. Moreover, it will take in account specific requirements of a given case study which poses a big challenge for any approach that may try to address the current security issues.

III. Proposal

This proposal has two main objectives. The first is to evaluate and compare well-known security algorithms to verify their performance and the second is to decide where decryption/verification should be placed in GINSENG's current topology. In order to accomplish the first one, four different algorithms will be compared by their run time and energy consumption:

(i) AES-128: AES is a symmetric key encryption algorithm, which uses keys of 128, 192 or 256 bits. However, only the 128 bits version will be evaluated as the hardware implementation is AES-128 and does not support longer keys. The block size is 128 bits.

(ii) 3DES: this algorithm is also a symmetric key encryption algorithm. However, it relies on simple DES applied 3 consecutive times, each with a different key and varying the operation (encryption, decryption and encryption once again). These keys can have 56 bits, 112 bits or 168 bits and the block size is 64 bits.

(iii) MD5: Message-Digest Algorithm 5 (MD5) is a widely used cryptography hashing function that given a certain input generates a 128 bits digest that enables the verification of a message after it has been transmitted over a WSN and that also can provide authentication.

(iv) SHA1: SHA1 is another cryptographic hashing function that generates a digest of 160 bits for a given input. Unlike MD5, the larger digest reduces the probability of hashing collisions.

In the specific case of AES-128 supported by hardware, no implementations were found across the research community that contemplate both encryption and decryption (or decryption only) for high level access (layers above MAC Layer such as the network layer). This is mostly a cause of the CC2420 datasheet statements, which redundantly state that only encryption is supported for direct access (standalone mode). However, as some authors such as Didla et al. speculated in their algorithms comparison work, there should be the possibility of doing both operations if the process of sending and receiving data is emulated (in-line mode is supporters with both operations for MAC layer security). This has been explored in this article and after some efforts to circumvent the limitations (eg. RX buffer is not writable unless a trick is applied) and the lack of documentation, a full implementation was achieved. This implementation uses the following steps to provide both encryption and decryption:

(i) Read security register 0.
(i) Ensure that no key is set and that in-line security is disabled in register 0 by using bit masks.
(ii) Select key 0 or key 1 through register 0 (already in memory) for the encryption/decryption operation.
(iii) Set in-line security mode to AES Counter Mode (AES-CTR) in register 0 to enable encryption and decryption operations.
(iv) Write security register 0.
(v) Set nonce for the counter block in big endian byte order.
(vi) Read security register 1.
(vii) Set TX/RX (encryption/decryption, respectively) frame header size to 0 in security register 1 as we are not interested in using complete frames but rather to use our own payload/data.
(viii) Write security register 1.
(ix) Write data to encrypt/decrypt to the TX/RX FIFO queue depending if the operation is encryption or decryption.
(x) Send strobe to encrypt/decrypt TX/RX FIFO queue contents.
(xi) Wait until the operation is complete.
(xii) Read total length of data in TX/RX FIFO queue.
(xiii) Remove automatically added footer from length (discard).
(xiv) Read encrypted/decrypted data from TX/RX FIFO queue.
(xv) Restore registers 0 and 1 to defaults.

Finally, regarding the second objective, all the tests will be carried out taking in account the positioning of the decryption/verification mechanisms. In the first case, such mechanisms will be placed at a sensor node – the sink node. In the second one, operations will be delayed and placed at the middleware level. Although it may look like that the difference of the decryption/verification positioning will not affect the system, it is a misleading idea. This can be better understood if we look deeper into the sink node's operation. In fact, this special node has no free slots, which for now means that it is always busy waiting for packets to arrive from other nodes and forwarding them through the serial interface to a computer, not having dedicated time to run security algorithms.

IV. Tests Scenario

As this article relies on an underlying case of study, the first step is to decide the tests scenario. This scenario should not only be representative from the industrial perspective of GINSENG project but also adjustable to the current configuration of GinMAC. As stated before, GinMAC is a TDMA MAC layer where the available time is divided in slots which are assigned to sensor nodes. As each sensor node has its specific slots to communicate, collisions may be avoided as also of most possible delays. Essentially, this MAC layer relies on a

static schedule that is pre-programmed in the sensor nodes, using a tree topology to support multi-hop routing. In Fig. 1, it is possible to observe the example of slots definition for a given sensor node with 2 children and 1 parent.

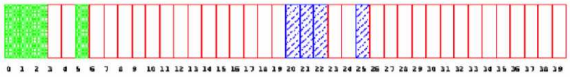

Fig. 1. GinMAC Slots Example

In this slot definition, the blank slots are slots when the node is sleeping, the green slots are the first slots to the normal transmission and the blue slots are the slots for the case when retransmission is needed. Basically, the first three slots of each group are upstream slots (one for the node itself, two for the children to send to the upper level) and the other is a downstream slot used for broadcast communication from the sink sensor node (the root node in the tree) to all the other nodes. Considering this, it is easy to deduce that if the schedule is well adjusted the delay requirements are met, retransmissions can occur and most of the packets will arrive and even more important the energy consumption decreases because most of the time the sensor node is in sleep mode (radio turned off).

In the current configuration, each epoch has the duration of 1 second, allowing 100 slots of 10 milliseconds each (2.5 milliseconds pre-processing time, 5 milliseconds processing time and 2.5 milliseconds post-processing time). These slots are then divided through the nodes allowing up to 16 processing slots, 76 active (downstream and upstream) slots and 8 sleep slots. This leads us to the network topology, which in this case is a 3-2-1 tree with 15 nodes plus the sink node, as observed in Fig. 2. Moreover, getting back to the positioning of the decryption/verification process, we verify that the algorithm should be capable of decrypt/verify hash in the first 2.5 milliseconds window of the slot to work properly as the processing time in the case of the sink will be needed to forward data through the serial interface. If not, as in most algorithms situation (to validate later), there may be not enough time to run leading to losses in the current and upcoming slots.

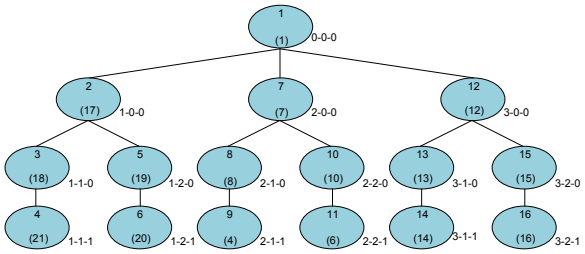

Fig. 2. 3-2-1 Tree

Also, in brackets it is included the mapping of real IDs to the tree topology, as a real scenario of deployment has been chosen and it has specific places to put the nodes. These places are well defined, and may be located when looking at Fig. 3 and Fig. 4, which contain the definition and also the distances involved.

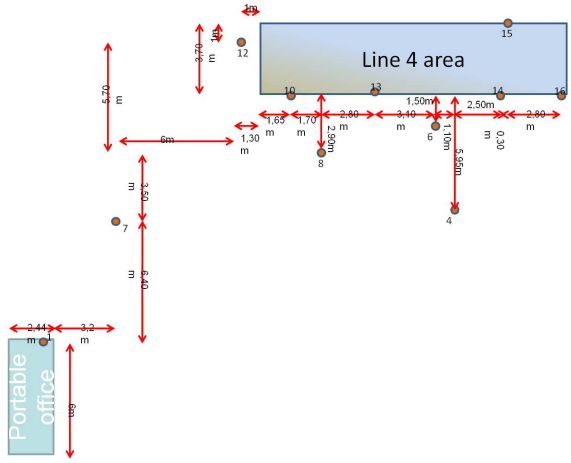

Fig. 3. Line 4 - 1

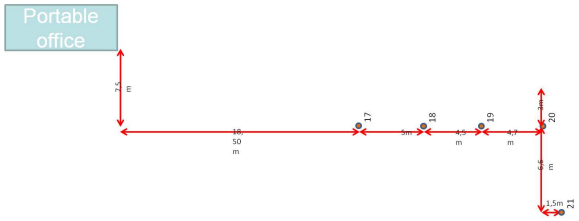

Fig. 4. Line 4 - 2

Regarding the nodes themselves, they consist of Crossbow TelosB motes [13] which feature a MSP430 Microcontroller (MCU) working at 8 MHz, a CC2420 radio transceiver, 48 KB of Read-only Memory (ROM) and 10KB of Random Access Memory (RAM). As operating system, currently they run Contiki Operating System (Contiki OS) [14] developed by the Swedish Institute of Computer Science (SICS), which provides full low-level support for GinMAC and also a multi layer (2 and 3) stack protocol called RIME [15] that is used to send messages across the network with 8 bits addressing (maximum of 256 nodes). Below GinMAC, the framing and radio transmission use the standard IEEE 802.15.4 [16], the 2.4 Ghz band using channel 16 and a bandwidth of 250 Kbps. GinMAC itself also provides layer 2 routing, which in GINSENG is done statically due to the referred topology.

V. Tests Methodology

In order to run a conclusive and valid study, tests specification plays a main role. Firstly, the traffic sources running in each node must be decided. In this particular case, nodes will run a simple application that gathers data from sensors (temperature, humidity, etc.) at a given rate and sends it through a single stream to the sink node (multi-hop is used at levels 2 and 3). The rate of sampling from the sensors will

also be the rate at which the nodes are sending data, and in this case it makes sense to test 0.25 Hz, 0.5 Hz and 1 Hz. Also, the packet payload size will vary during the tests, being 16 bytes or 32 bytes. Secondly, the last scenario specification is related with the evaluation parameters. Depending on the type of the test, the following parameters will be evaluated:

(i) Average end-to-end delay measured in milliseconds.
(ii) Average end-to-end packet delivery ratio in percentage.
(iii) Average time used by each security algorithm to run a given operation in milliseconds.
(iv) Average CPU energy consumption during each operation of each security algorithm in microwatts/hour.

Therefore, we will have a first set of tests that evaluates the time and energy used by security operations of each algorithm (either with 16 or 32 bytes payload) and then, depending on the results obtained and its feasibility in the system, present results of packet delivery ratios and delays when comparing to the baseline software and using different send rates and payload sizes.

Also important is the methodology used for the evaluation of the CPU energy consumption, which in this case is obtained from a Contiki module called Energest. This module works simply by storing the number CPU cycles for each operation and then, using pre-determined power consumption values for a single CPU cycle, obtains the total power consumption of the particular operation.

Finally, last but not least important is the tests duration. As this system does not change a lot during the time due to its deterministic character, 6 minutes is enough. This value has been chosen due to several factors, but mainly to allow the tree and the radio communications to stabilize, to allow a representative sample and to contemplate 30 seconds of warm up and cold down periods which will not be taken in account due to overheads from the protocols involved. Even so, each test will be carried out 5 times to ensure good confidence intervals in the obtained values.

VI. Experimental Results

A. Encryption/Decryption Algorithms

The first test is related with the encryption/decryption algorithms, comparing the time and power spent by each of them to encrypt/decrypt 16 bytes of data. Data regarding this test is presented in bar graphs in Fig. 5 and Fig. 6.

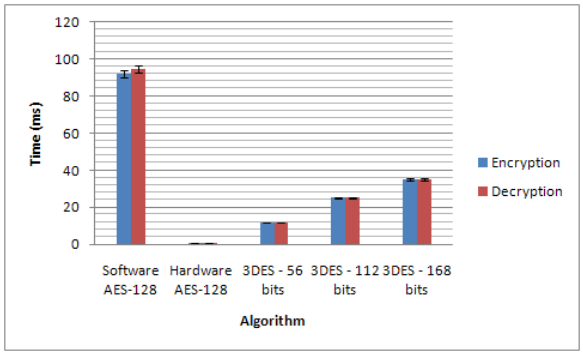

Fig. 5. Encryption and decryption running times with 16 bytes payloads

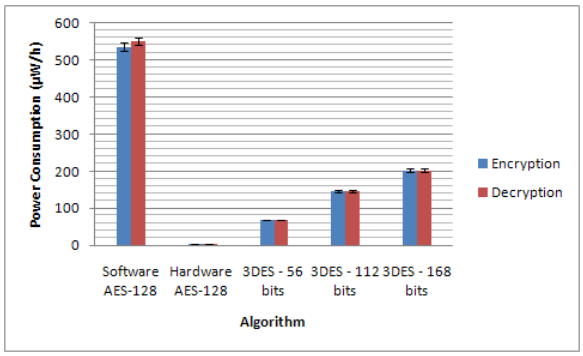

Fig. 6. Encryption and decryption power consumptions with 16 bytes payloads

The second test it is almost like the first but uses 32 bytes payloads as reference, being represented in Fig. 7 and Fig. 8 for both running time and power consumption, respectively.

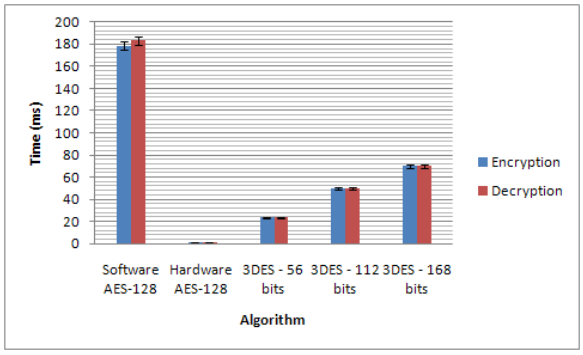

Fig. 7. Encryption and decryption running times with 32 bytes payloads

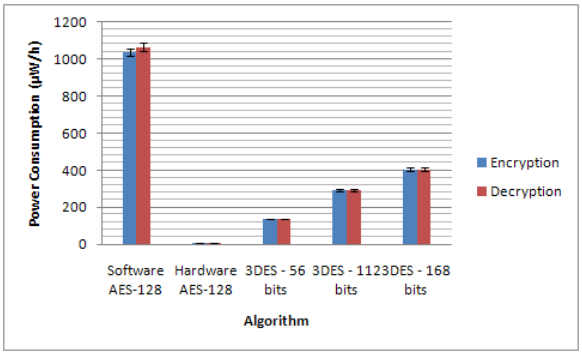

Fig. 8. Encryption and decryption power consumptions with 32 bytes payloads

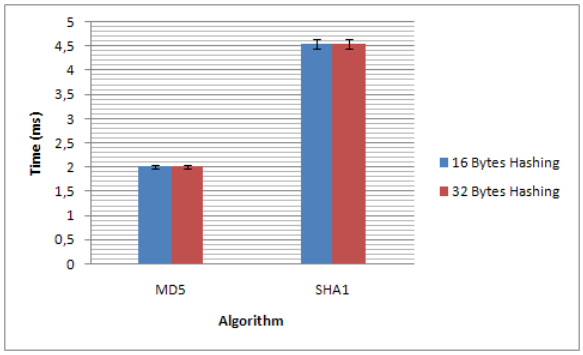

Fig. 9. Cryptographic hashing algorithms running times

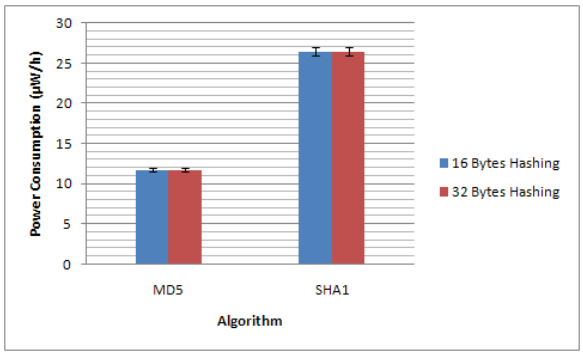

Fig. 10. Cryptographic hashing algorithms power consumptions

As illustrated in the above graphs, most algorithms take a long time to run. Specially, software AES-128 wastes almost 1/10 of an epoch to encrypt/decrypt data, turning it a real bad choice to use in real-time environments as it affects the epoch structure. In fact, the duration of the epoch is not a random value. It is chosen so that the upper boundary of the delivery delay is 1 second (one epoch) according to the requirements defined by the security and performance board of the oil refinery used as test scenario. Thus, if this epoch is compromised, the whole delivery assured delays are also compromised and the requirements for these kind of environments are not met. Also, regarding the difference between encrypting and decrypting, only software AES-128 takes different times to do both operations. Moreover, considering that a node can take at most 8 ms to encrypt and decryption at the sink cannot take more that 2.5 ms, only hardware AES-128 is eligible for being applied in GINSENG, as it takes less than a millisecond to run. Lastly we my also observe that using 32 bytes payloads duplicates the time and energy consumption in all the cases. Thus, the conclusions taken with 16 bytes payloads remain valid and become even more supported. On the other hand, hardware AES-128 remains valid and using less than a millisecond to run. Regarding energy, as it is correlated to time because it was measured with software, the same conclusions can be taken as the ones referred above.

B. Cryptographic Hashing Algorithms

Other relevant tests were the ones regarding cryptographic hashing algorithms. Below, some graphs present the collected data in Fig. 9 and Fig. 10.

From the graphs, and taking in account the time constraints of GINSENG referred in the previous sub-section, the first conclusion is that only MD5 may be applied. In fact, SHA1 takes twice the time and energy to run and generate a digest that is not twice as big. Thus, SHA1 is not a valid option. Also, an interesting fact is that both algorithms take exactly the same time and energy to create digests for either 16 bytes or 32 bytes payloads.

Taking in account the previous results and the obtained conclusions, one may deduce that only hardware AES-128 and MD5 algorithms should be considered in the last tests. The main reason to this is the strict time constraints of GINSENG and GinMAC being a TDMA MAC Layer. Actually, if other algorithms were included in the last tests, the whole TDMA schedule would be compromised and it would be impossible to obtain any sort of results.

C. Decryption/Hashing Verification Placement

As mentioned before, only hardware AES-128 and MD5 were considered for the last tests. In order to compare them with the system without security, normal communication was established at different rates and with different payloads. With such setup and the specified runtime for each test, data was gathered from the received packets at the sink. Fig. 11 and Fig. 12 illustrate that data.

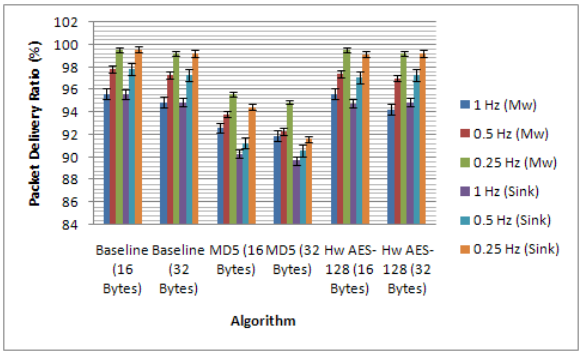

Fig. 11. Average packet delivery ratios with and without security algorithms

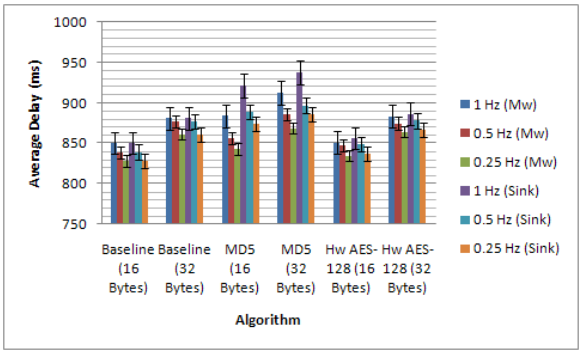

Fig. 12. Average delays with and without security algorithms

From the above graphs, it is possible to notice that using bigger packet sizes increases the delay in the network but without putting in risk the delay constraint of 1 second in any case. Then, we may also observe that hardware AES, due to its low resources utilization, does not affect the performance of the network, either regarding the average delay or the average packet delivery ratio. On the other hand, MD5 causes losses that may not be tolerable in some scenarios and also a slightly significant increase of delay that may be discarded when referring to the 1 second constraint of GINSENG. Moreover, overall losses in all cases are not affected in a relevant way when varying the packet size, mostly because an higher packet size usually means more time to transmit or retransmit. However, we are not taking in account integrity errors, which may also be relevant. Finally, it is also possible to verify that increasing the sending frequency poses an overall performance reduction. This is mainly related with the fact that the slots become more occupied, the spectrum more used and the fault tolerance decreases as retransmission slots are each time more used and sub-dimensioned. Even though this fact, we still have good performance indicators in most cases.

VII. Conclusion

From the previous section, two main conclusions may be drawn. First, only hardware AES-128 and MD5 may be applied in GINSENG without affecting the normal behavior of the system. Then, the placement of decryption/hashing verification should be placed at the middleware in all the cases that it is possible. However, placing it at the sink node does not affect the performance in a very significant way (if we tolerate about 10% losses in some cases) or a little bit more delay without compromising the 1 epoch delivery (1 second). In conclusion, preliminary evidence was collected regarding which security algorithms to apply in GINSENG. Still, more tests need to be carried out and new ways of dealing with the issues should be found.

VIII. Future Work

Taking the previous section last conclusion, future work should be planned in order to solve some possible issues and to try to obtain a replacement for MD5. For instance, CC2420 hardware capabilities may be explored to generate Message Authentication Codes (MAC) with modes such as AES-CBC or AES-CCM but with direct access (not only for real inline mode use).

Furthermore, issues like the key distribution, key renewal times and also security levels should be evaluated and discussed. In fact, finding good solutions to solve them is a challenge that has not been overcome yet without compromising the tiny resources available in WSNs.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement N° 224282.

References

[1] C. J. Sreenan, "GINSENG - Performance control in Wireless Sensor Networks," University College of Cork, Cork, Ireland, Sep. 2008. [Online]. Available: ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/necs/fp7-fact-sheet-ginseng_en.pdf

[2] T. O'Donovan, J. Brown, U. Roedig, C. J. Sreenan, J. do Ó, A. Dunkels, A. Klein, J. S. Silva, V. Vassiliou, and L. Wolf, "GINSENG - Performance control in Wireless Sensor Networks," in *Proceeding of the 7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, Jun. 2010, pp. 1–3.

[3] J. B. Schmitt and U. Roedig, "Worst case dimensioning of Wireless Sensor Networks under uncertain topologies," in *Proceedings of the 3rd IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, (WiOpt'05) Workshop on Resource Allocation in Wireless Networks*, Riva del Garda, Italy, 2005.

[4] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*, 1st ed. Springer, Mar. 2002.

[5] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication 197, Nov. 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[6] Y. W. Law, J. Doumen, and P. Hartel, "Survey and benchmark of block ciphers for Wireless Sensor Networks," *ACM Transactions on Sensor Networks TOSN*, vol. 2, no. 1, pp. 65–93, Feb. 2006.

[7] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A link layer security architecture for Wireless Sensor Networks," in *Proceeding of the 2nd International Conference on Embedded Networked Sensor Systems*, Baltimore, MD, Apr. 2004, pp. 162–175.

[8] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, MIT Laboratory for Computer Science, Cambridge, MA, Apr. 1992. [Online]. Available: http://tools.ietf.org/html/rfc1321

[9] National Institute of Standards and Technology, "Secure Hash Standard (SHS)," Federal Information Processing Standards Publication 180-2, Aug. 2002. [Online]. Available: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[10] J. Granjal, R. Silva, E. Monteiro, J. S. Silva, and F. Boavida, "Why is IPSec a viable option for Wireless Sensor Networks," in *Proceedings of the Fifth IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, Sep. 2008, pp. 802–807.

[11] Chipcon, "CC2420 Datasheet," Jun. 2004. [Online]. Available: http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf

[12] S. Didla, A. Ault, and S. Bagchi, "Optimizing AES for embedded devices and Wireless Sensor Networks," in *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, no. 4, 2008, p. 10.

[13] C. Technology®, "TelosB™ Datasheet," Dec. 2010. [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf

[14] Swedish Institute of Computer Science, "Contiki os," Jan. 2012. [Online]. Available: http://www.contiki-os.org

[15] A. Dunkels, "Rime — A Lightweight Layered Communication Stack for Sensor Networks," in *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, Delft, The Netherlands, Jan. 2007.

[16] Institute of Electrical and Electronics Engineers, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LRWPANs)," IEEE Std. 802.15.4, Oct. 2003. [Online]. Available: http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

Annex C – WSNOpenSec - An Interoperable Security Model for Wireless Sensor Networks

ELSEVIER

Pervasive and Mobile Computing

WSNOpenSec - An Interoperable Security Model for Wireless Sensor Networks

André Gomes[a]*, Jorge Granjal[a], Jorge Sá Silva[a]

[a]*University of Coimbra, Pólo 2 - Pinhal de Marrocos, 3030-290 Coimbra, Portugal*

Abstract

Wireless Sensor Networks are evolving at a high rate, with researchers focusing their work on the several problems and challenges that this kind of networks have. However, security is still not developed at a level that ensures trust and makes possible the creation of standards. Current implementations are focused and developed to specific architectures or even closed specifications associated with standards from industrial consortiums. Together with these limitations, existing proposals also face the problem of narrow coverage of security aspects. This paper intends to describe a new model named WSNOpenSec, which proposes solutions to some of those limitations while being open source for the community.

Keywords: "Criptography"; "Interoperable Security Model"; "Key Distribution"; "Security in Wireless Sensor Networks".

1. Introduction

Security in Wireless Sensor Networks (WSNs), although it is a topic which already has a lot of proposals and studies, is still in a very embryonic state without meaningful solutions that can be widely adopted and standardized. This last observation has one major cause – security in this kind of networks is not as simple and easy to implement as in traditional networks. This can be explained by some factors that are inherent to WSNs. For example, motes are typically powered by AA batteries, which under high resources utilization tend to have a very low autonomy. And high resources utilization can be a symptom of security itself caused by another inherent factor of WSNs: cryptography and related algorithms are designed to modern computers in order to provide a high level of security, meaning that the resources utilization is also high and not adapted to constrained devices such as WSNs motes (8MHz Central Processing Unit (CPU), 10KB of Random Access Memory (RAM), and others as explained in next chapter). Also, as referred before, most of the scenarios with high security requirements are not typical scenarios but critical scenarios where factors such as real-time operation, integrity, long battery autonomy and reliability are mandatory.

Another important gap left by current solutions for WSNs security concerns compatibility and interoperability. Most proposals from the research community tend to focus in a specific platform with a specific OS, which can be explained by the difficulty of finding solutions that can be widely applied to several architectures and to different OSs designs. This difficulty is not only from the technical point of view of security, but also a consequence of the

* Corresponding author. Tel.: +351 239 790 000; fax: +351 239 701 266.
E-mail address: asng@dei.uc.pt

lack of standardization and normalization in the communication stacks for WSNs, which nowadays is being tackled by the Internet Engineering Task Force (IETF) with efforts to deploy an adapted Internet Protocol (IP) stack for WSNs.

Bearing these last paragraphs in mind, and considering that a big influence to this paper is an European project (on which the authors of this paper have participated) that intends to apply WSNs in critical scenarios, the need for a security model that can both be applied to WSNs without compromising their normal operation and that is compatible and interoperable between different architectures and OSs emerges as a necessity.

Therefore, this paper proposes a security model that can be applied to a whole range of different architectures/equipments without compromising their normal operation and assuring that communication is done in a secure way. Moreover, it is intended that the proposed security model is portable for multiple architectures, lightweight, easy to use, transparent for the common programmer and modular so that it can be configurable and able to adapt to any possible scenario of use.

Finally, this paper is structured as follows. In section 2, a comprehensive overview regarding the current proposals to address security in WSNSs is presented. In section 3, the proposal itself is described and its objectives and possible applications are also explained. In section 4, the validation tests for the proposed security model are defined and described with results to support them. Finally, section 5 concludes this paper and presents challenges and problems which are still to overcome in future work.

2. Related Work

2.1. IEEE 802.15.4 Security

IEEE 802.15.4 [1] already implements several security features [2][3], which then can be used at the MAC Layer to enable layer 2 security.
Essentially, the encryption/authentication algorithm to use is set when data is transmitted and, when data is received, the decryption/verification algorithm that has been used is defined a specific header for the receiving sensor node to be able to process it.
However, this standard as a big flaw – it does not specify how keys should be distributed, leaving this process to upper layers that can be dealt with technologies such as ZigBee.

The encryption/authentication algorithm specified in the standard is AES (Advanced Encryption Standard) with a 128 bits key length (16 Bytes). This is an important point, as having only one algorithm leads to the possibility of having hardware support for its operation, which is a big advantage to constrained devices such as sensor nodes.

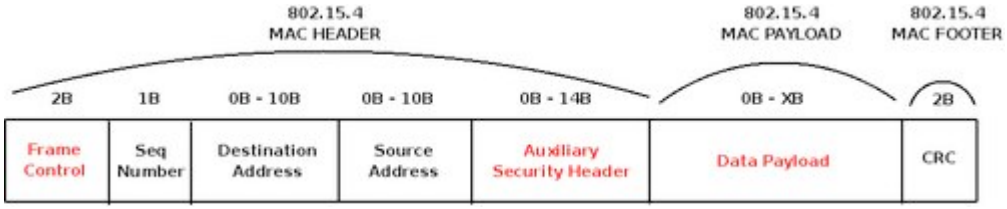

Figure 1 - IEEE 802.15.4 Frame [35]

The AES algorithm is not only used to encrypt the information but also to check and validate the data which is sent (data integrity), to authenticate nodes and to assure non-repudiation. This concept is achieved using a Message Integrity Code (MIC) which can also be named as Message Authentication Code (MAC). This code is appended to payload in the frame, and provides assurances of both the MAC header and the payload. However, it also causes some overhead to the communication that can be harmful.
The code is generated by encrypting parts of the MAC frame using a given 128 bits key that is shared by all the sensor nodes, leading to a reverse verification at the destination and therefore a comparison between embedded MIC

and generated MIC using the key at destination. If the comparison is not a match, it is possible to assume that data integrity is not assured or that the sensor nodes that sent it are not part of the network (authentication concept). If different keys are used and maintained for each specific node, while verifying it is also possible to assure the non-repudiation concept (100% certain that a frame came from a given sensor node).
These MIC codes can have different sizes: 32, 64, 128 bits, however they are always created using the AES-128 algorithm with different modes of operation. Its size corresponds to the number of bits that are going to be attached to each frame, and like hashing functions the larger the number is, the more secure the verification is (less collisions). Hence, there is the need to find a tradeoff between security and the overhead that a larger size causes on the frame.
In terms of data security, it is performed encrypting the data payload field with the same 128 bits key but without generating any overhead. In fact, even the if the payload does not match the block size of AES-128, the hardware module must support padding and therefore it deals with blocks of any size without the need to increase the size.

2.2. Zigbee Security

ZigBee [3] [4] implements two extra security layers above IEEE 802.15.4: the Network and Application security layers. All the security policies also use the same algorithm as IEEE 802.15.4 (AES), so the embedded hardware support that is feasible for IEEE 802.15.4 may also be used. There are three types of keys:

- ✓ Master Keys: They are installed offline in each node. Their function is to keep confidential the Link Keys exchange between two nodes in the Symmetric-Key Key Exchange (SKKE).
- ✓ Link Keys: unique keys between two adjacent nodes. These keys are managed by the application level, leading to more resources utilization. Hence, they are often not used.
- ✓ Network key: It is a unique 128 bits key shared by all the devices in the network. It is generated by the Trust Center and renewed according to a defined policy. Without this key, a given node cannot join the network. This creates a mandatory requirement: the key must be pre-programmed to start the first authentication process. Every time the trust center orders a Network Key renewal, the new one is spread through the network using the old Network Key (see figure 8) to keep it safe from attackers. Additionally, as the new key starts being used, the Frame Counter parameter is initialized to zero.

Each pair of devices can have defined the Network and Link Keys. In this case, the Link key is always used which increases the security level but also uses more resources. There are two types of security policies which the Trust Center can follow according to the Zigbee standard:

Commercial mode: the Trust Center shares Master and Link Keys with any of the devices in the network according to the example in figure 2. This mode requires high memory resources (powerful devices), and therefore offers a complete centralized model for the Key Security control.

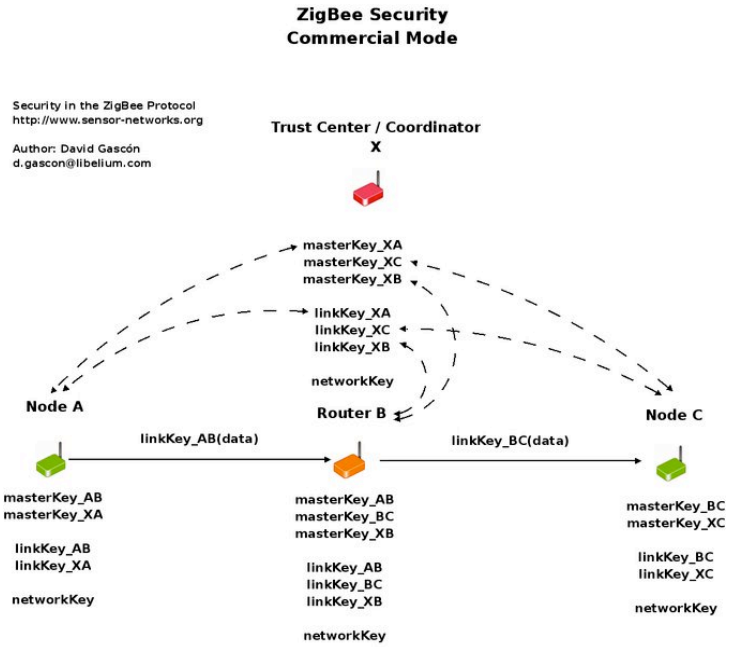

Figure 2 - ZigBee Commercial Mode Security [35]

Residential mode: the Trust Center only shares the Network Key (more useful and feasible for devices with high constrains). This is the mode that is usually chosen for the WSN topology exemplified in figure 3.

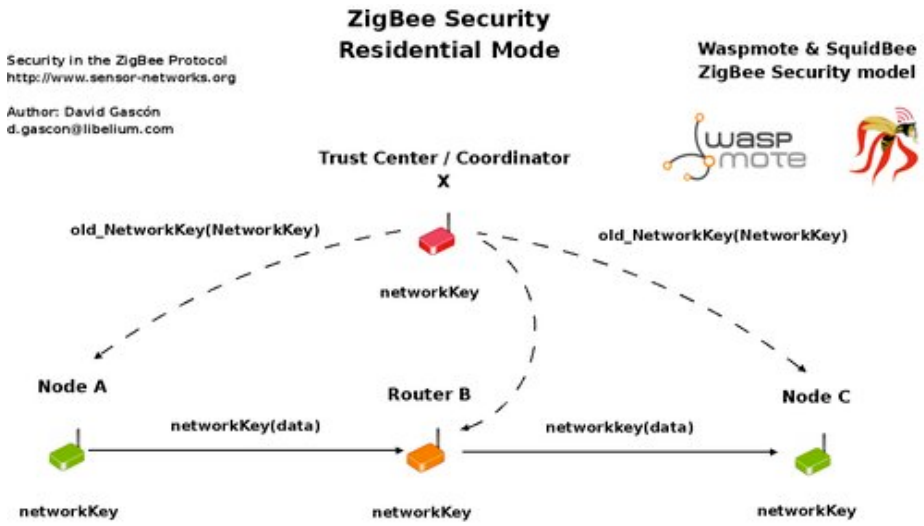

Figure 3 - ZigBee Residential Mode Security [35]

In conclusion, ZigBee provides a stable security model with some extended features that bring security to upper layers (as opposed to only IEEE 802.15.4). However, and besides the fact it is a closed and commercial technology, it does not put security below the MAC layer, does not provide any information or support regarding security levels, and it is not compatible with other communication stacks/architectures than the ones it was developed to.

2.3. WirelessHart Security

The WirelessHART technology [5] was designed to enable secure industrial wireless sensor network communications while ensuring ease-of-use is not compromised. In fact, its security mechanisms are so embedded in the whole technology that they cannot be disabled.
As ZigBee and 802.15.4, it also relies in AES-128 but with the concept of end-to-end sessions in order to assure that only the nodes communicating can know the contents of the payload and use that data successfully.

Risk Assessment / Reduction
According to WirelessHART's consorcium, an attacker is only relevant if it has access and enough knowledge/motivation. The security architecture embedded in WirelessHART addresses these three key points with the following actions:

- ✓ Minimize, control, and audit access.
- ✓ Require high levels of technical expertise to subvert.
- ✓ Reduce the consequences (span and duration) of any individual security breach.

WSNs security can be divided into two main categories, which can be named Data Protection and Network Protection. The first intends to maintain privacy and integrity of data transmitted along the network, while the second aims to assure the functionality of the network itself in case of internal and/or external attacks that can be provoked intentionally or not.

Data Protection
The privacy related security features implemented by WirelessHART aim to prevent eavesdropping by unauthorized devices both inside or outside the WSN. By default, a WSN with WirelessHART provides end-to-end CCM mode of AES-128 at the network/transport layer. In addition to these individual session keys, and similar to IEEE 802.15.15/ZigBee, there is a shared and common network key for all the devices in the network that eases the broadcast activity as it is needed.
The rotation of these keys (key renewal) is assured to provide an higher level of protection. However, it is not adaptable which means that they are changed automatically but according to a pre-defined security policy.
Another separate 128 bits key is also used. This is a pre-configured key intended to be used during the joining process of a node while keeping it private. Moreover, the join key also serves as authentication to the security manager that the device belongs to this network. Their policy is also different. They are treated separately from the other keys to enhance security, and can be unique to each device or common to the whole network according to the defined security policies.
Also relevant are the integrity related security features. These ensure that data sent over the WSN has not been tampered or falsified along its route. The mechanism is very simple. An integrity check field is computed and added to the packet. After the destination node receives it, it calculates the value again and by comparing to the one embedded in the packet it can deduce if the contents have not changed. In the case of WirelessHART, an advantage is that not only the payload is protected but the routing information is protected as well. Thus, attacks that aim to change routing are prevented and avoided.
Finally, data integrity also involves verifying that a certain packet has come from the correct source (non-repudiation). It works by verifying the integrity check field and the information used to generate. In fact, as the session key is unique, a certain node can verify that a given packet has came from the right sender by comparing the signature in the integrity check field (generated using that unique session key).

Network Protection
A WSN in general also needs tools to protect it against attacks. These attacks can attempt to compromise the network by inserting Trojan horse devices, impersonating networks to get sensitive data from legitimate devices, and disrupting the network to deny service. Attacks can be launched from outside or inside the company by external people or employees. Network security depends upon techniques to support Authentication, Authorization, and Attack Detection (AAA).

A WirelessHart gateway and the sensor nodes which will join its network must be configured to control which devices can access the network, because the first principle to ensure security is that all the devices respect the policies and maintain the security level. Therefore, the gateway has a defined authentication process which it uses to negotiate the join of new devices by ensuring they are legitimate and authorized to access the network. This join process, as everything in WirelessHART, is encrypted end-to-end from the node to the gateway.
Finally, to prevent DoS attacks that may try to jam the radio or overload processes such as packet acknowledgements, WirelessHART devices are capable of detecting anomalous conditions such as high levels of traffic and high number of retransmissions in order to notify an operator and solve the existing issue.

Device Roles
Devices are network routers as well as data sources, meaning that they can both forward data in a multihop topology and at the same time gather data by themselves. However, they are not authorized to play the roles attributed to the gateway as it has a specific signature that ensures that. Also important to refer is that WirelessHART does not implement the TCP/IP communication stack and therefore is explicitly safe from many typical attacks related with IP.

Security Manager
As discussed, Join, Network and Session Keys must be provided to the WirelessHART Network Manager and Join keys must be provided to Network Devices. These keys are used for device authentication and encryption of data in the network. Also as referred, the WirelessHART Security Manager is responsible for the generation, storage, and management of these keys which have different policies according to their use and the global security policy that is being enforced at the location.
There is one Security Manager associated with each WirelessHART Network. The Security Manager may be a centralized function in some plant automation networks, servicing more than just one WirelessHART Network and in some cases other networks and applications.
Transitional WirelessHART Device States:

- ✓ Idle - The device is quiescent and its wireless transceiver is not active. It has no knowledge of the WirelessHART network.
- ✓ Joining - The device is listening for the network, attempting to acquire an advertisement and requesting admission to the network.
- ✓ Quarantined - The device has successfully joined the network but only has a security clearance to talk with the Network Manager. It is not available or allowed to perform data acquisition or control functions or otherwise communicate with the Gateway.
- ✓ Operational - The device can be accessed by Host Applications via the Gateway. It is integrated in the system's operation.

Additionally, a device may also be Suspended or enter a Re-synching state after joining.

In conclusion, WirelessHART security is an industry leading technology with mechanisms that ensure that attacking a WSN with it is not easy and even if the attack is succeeded it will be minimized and detected. However, it has four major disadvantages:

- ✓ It is paid and contains a closed specification, meaning that is not the right choice for common setups and cannot be modified accordingly.
- ✓ Being closed, it is also not modular or interoperable, which can seriously affect its use across different platforms.
- ✓ It does not provide mechanisms to allow a dynamic key refresh rate based on security indicators gathered from the network.
- ✓ It lacks IP, which is simultaneously an advantage (does not have to deal with IP specific threats) but also a major disadvantage because once again interoperability and compatibility are not a reality.

2.4. Other Related Work

During many years, RC5 proposed by Rivest, R. and Skypjack introduced by NIST have been considered by the research community as the most relevant and suitable security algorithms for WSNs when it comes to symmetric block ciphers [6]. However, more recently technology has evolved and a new algorithm has taken its place as the most adopted and secure approach without compromising the low resource requirements of WSNs - Advanced Encryption Standard (AES) [7]. AES was initially published by Daemen, J. and Rijmen, V. as Rijndael in 1998, but with the selection process required for standardization it only came out as AES in 2001 [8]. Later, researchers have developed implementations for WSNs with optimizations and today it is considered the most preferable option [9]. Therefore, when it comes to evaluate possible security algorithms AES has to be considered as the best candidate for data protection with symmetric block ciphers, but others should not be taken apart from any comparison study. However, as AES took the main role for data protection, most of the current days embedded radios for WSNs already have its cipher implemented using hardware which leads us to great results that cannot be matched by software implementations of other algorithms in terms of speed and resources utilization.
Also, it is important to evaluate algorithms that are capable of providing integrity, non-repudiation and authentication without requiring powerful computational resources to run. In this area, industry already uses algorithms such as Message-Digest algorithm 5 (MD5) [10] and Secure Hash Algorithm 1 (SHA-1) [11] that can meet these requirements.
Regarding security algorithms comparison, there are already some proposals that try to evaluate well-known and proven implementations to general scenarios without specific requirements. However, these comparisons usually do not take in account that some platforms already have hardware based security modules and that even when a security algorithm seems to fit the platform resources it may not fit a specific application at all because of its specific requirements. For instance, Granjal, J. et al [12] evaluated algorithms such as Secure Hash Algorithm 1 (SHA1), Secure Hash Algorithm 2 (SHA2), AES and Triple Data Encryption Standard (3DES) but did not try different key sizes neither some available hardware implementations of AES available in radio transceivers such as the Chipcon CC2420 [13]. Others, like Didla, S. et al [14], extended their study past the software and also compared the hardware approach but didn't use its full potential as they let the decryption process out of the study.

3. WSNOpenSec

3.1. Overview

The proposed security model, WSNOpenSec, is both multi-layer and cross-layer as it can be defined as a new layer of abstraction in the communications stack. In fact, as may be observed in figure 4, it gathers and sends information to the 5 layers of the TCP/IP model with a set of modules. With this architecture, the main objective is to provide an API with direct access for the layers which can be used in a transparent way and enables security for the complete stack. At the same time, the security layer is also capable of getting information from each of the other layers and may adjust security levels according to the values of defined metrics.
This adjustment is rather important to WSNs, as depending on a whole range of factors a given node may lower the security level to save resources or to prioritize other aspects of communication.
Finally, regarding the interoperability and compatibility aspect, this model relies essentially on the use of a standardized IP stack (6lowPAN). However, other aspects are also required to assure such requirements. For instance, the code must be portable for any architecture with a C compiler, ranging from 8 bits architectures to 32 bits architectures.

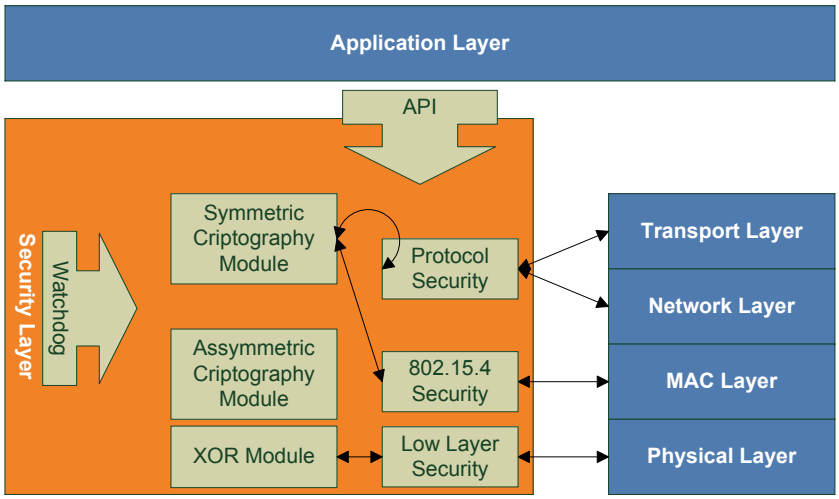

Figure 4 - Communication Layers with Security

3.2. Objectives/Requirements

The proposed model takes in account several objectives to bring innovation and a relevant scientific contribution to the existing solutions. These objectives are as follow:

- Provide a fully secure model for general use in communication stacks or standalone operations in WSNs;
- Allow the programmer to access the security layer in a transparent way through the use of a system independent API;
- Ensure that the security layer is portable for any architecture with an available C compiler (8 bits, 16 bits and 32 bits);
- Enable compatibility between different operating systems when the communication stack is the same;
- Enable interoperability with different types of sensor nodes by using a limited amount of memory, leading to a lightweight implementation (2KB of RAM and 8KB of ROM at most);
- Provide modularity to ensure that any platform is able to run the basic functions by disabling the not supported advanced features;
- Allow the definition of security levels and their dynamic change according defined metrics besides the default definitions already provided;
- Provide secure ways of distributing keys without compromising the normal operation of the sensor nodes by using efficient algorithms with optimized implementations;
- Add support for automatic key renewals based on the defined security levels or commands sent through the sink node.
- Add a contribution for the open-source community so that it has access to security mechanisms that currently are only available in closed/paid specifications.

3.3. Layers Components

Physical Layer

At the physical layer, there are not relevant proposals to provide protection against attacks such as man in the middle or denial of service. Although it may not be possible to directly protect nodes against denial of service at the physical layer, the security layer should provide mechanisms to detect the jamming and therefore allow the network layer to route the traffic through another available path.

By the contrary, to avoid attacks at the physical layer such as man in the middle, there is a simple mechanism that the security layer must implement and allow the programmer to use: introduction of low level protocol "disguise" in the signal that only other sensor nodes which are authorized can process. This is obtained at this security model through the use of the XOR module (explained later in this section). With this, it becomes much harder for an attacker to get the communication protocol and start any other type of attack. However, it adds overhead and requires more resources, which in the case of WSNs increases power consumption and becomes a drawback if the right trade-off between the security level and power is not found.

MAC Layer
At the MAC Layer, as there is the IEEE 802.15.4 open standard, this model does not bring a new implementation. However, as some operating systems (Contiki for instance) do not have support for the security mechanisms of IEEE 802.15.4 even though its set of MAC Layers is compatible with the standard, the security layer enables a whole API for the IEEE 802.15.4 security mechanisms that makes easy to add security to the already existing MAC Layers without explicit security support.

Network Layer/Transport Layer
At the network layer, this security layer provides a full set of functions to ensure that the IPv6 protocol (6lowPAN) has a tiny security header with only 2 bytes with the following contents:

- ✓ 1 byte to define how many bytes of the packet are encrypted after the security header.
- ✓ 1 byte with the following bits attribution:
  - 1 bit to select the key (key 0 or key 1);
  - 6 bits to specify the header size (if encrypted, 0 if not encrypted);
  - 1 bit for future use;

At the same time, using security algorithms for data confidentiality, data integrity, authentication and non-repudiation, it is assured that the network layer is secured and able to communicate with other network layers as long as the communication stack has the security layer defined in this proposal.

Application Layer
At this layer, as it is mostly controlled by the user, the security model only provides a set of basic functions to use in a standalone way in order to do operations which may not even be related with communication. These functions can be accessed the same way the communication layers do – an API.

3.4. Layers Communication

One key aspect when designing any security model is the communication between layers in a communication stack. In fact, security can be classified as end-to-end security or hop-by-hop security. In the first case, secure communication is established between two points and other nodes which relay the messages should not be aware of their content. In the last case, it is the opposite as every node in the path from source to destination should be able to decrypt the contents of the message.
Bearing this in mind, the security model proposed in this thesis provides a hybrid approach as other relevant proposals described in the state of the art do. This approach can be viewed in Figure 5.

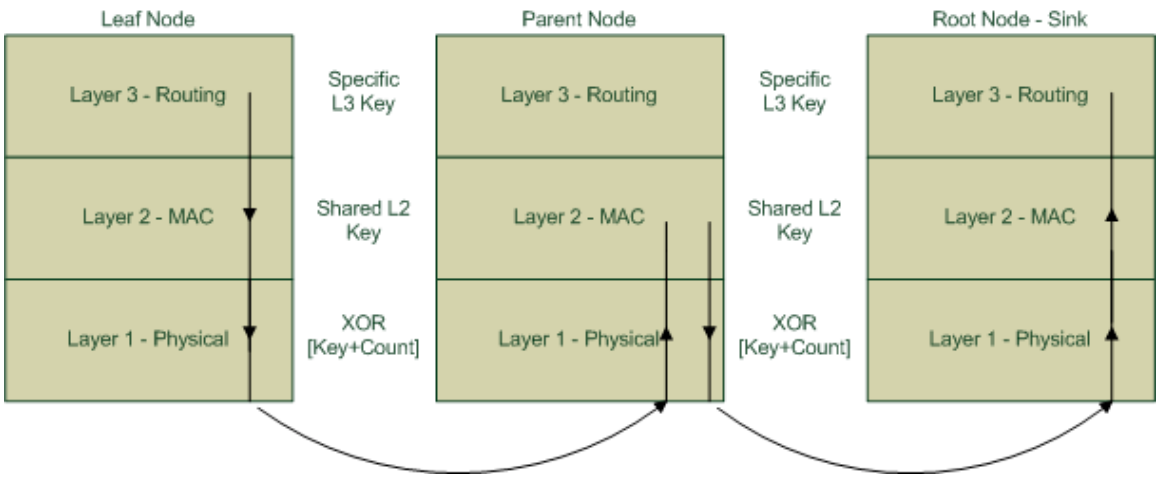

Figure 5 - Multihop Secure Communication

In this figure, it is considered that the leaf node is sending data to the root node (also known as sink node). However, as there is no direct path, data has to pass through the parent node so that it may reach the destination (typical multihop scenario). To achieve this communication with security, data is encrypted/signed with a specific layer 3 key established between the leaf node and the root node (only these nodes can get the decrypted data) and also with a shared layer 2 key that is generic for every node in the network (enables routing as frame and packet headers need to be decrypted). Additionally, a XOR process based on the layer 2 shared key is also done at the physical layer to protect the communication protocols. The arrows in the picture describe this process, but only as far as the security model goes (unsecure/application related security are not shown).

3.5. Symmetric-key Cryptography

Symmetric-key cryptography is a class of algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of cipher text. The keys can be the same or there should be a simple process to achieve one key from the other. These keys basically represent a shared secrete between two nodes that is used to keep a “tunnel” with private information.
Symmetric-key cryptography is also known for not being computationally intensive in general, but when comparing to asymmetric-key cryptography it has a major drawback: keys have to be shared between the nodes. This leads to some security problems and creates the need for an effective key exchange mechanism.

In terms of adopted algorithms, the choice for confidentiality was the hardware based AES-128. However, if there is no hardware support, there should be used a software algorithm. In this case, it is a hard choice. Although 3DES with a small key could be a logic choice, it is not very secure due to its inherent characteristics. This led to another approach: optimize the AES-128 software algorithm. In fact, Brian Gladman [15] developed a version with such optimizations that the performance can increase to a point where the algorithm runs in a time under 1-2 milliseconds as supported by Didla, S. et al [14]. This meets the requirements of this security model, leading to a double option for confidentially assurance in this model: AES-128 hardware and AES-128 software.

Finally, regarding authentication and integrity verification, the option for a software based algorithm fell on MD5. As hardware support is concerned, AES-128 with CBC mode or CCM mode also deal with the requirements for integrity, authentication and non-repudiation.

3.6. Asymmetric cryptography

Asymmetric cryptography refers to a system that requires two separate keys: one secrete key and one public key. There can be considered three types of asymmetric cryptography systems: public key cryptosystems, public key distribution systems and digital signature systems.
The first one allows secure communication between two nodes using algorithms such as Ron Rivest, Adi Shamir and Leonard Adleman algorithm (RSA)
The second one allows two nodes to securely agree on a shared secret, even if all their communications links are compromised. The shared secret generated from this process is then typically used as the key in a symmetric cryptography system.
Finally, the Digital Signature Algorithm (DSA) is the most widely used digital signature system. It can generate a digital signature, but has no privacy features and only enables the verification of a given entity.
As explained, these algorithms rely on a key pair (public and private key). These keys are mathematically linked, and the private key is extremely difficult to obtain from the public key. This is due to the mathematical relationships involved (the most notable ones being the integer factorization and discrete logarithm problems) that have no efficient solution.
Its process is relatively simple. The public key is used to transform the message into a form that cannot be decryptable without the matching private key. Therefore, any node which advertises its public key enables any other node to produce messages that can only be read by it because it is the only one with the private key.
However, asymmetric cryptography has a major drawback: it is very computationally intensive. Considering this issue, this model will only apply it for key distribution. After the key exchange process, the less resource intensive symmetric cryptography will be used.

3.6.1. Key Distribution

Key distribution is a fundamental topic when defining a security model than can provide a high level of security together with low resources utilization. A well known method to establish keys securely is the use of the above described asymmetric cryptography, only taking in account that asymmetric cryptography tends to be resource intensive.
However, if there is the requirement that only certain nodes may join a given network and start the key exchange procedure, other security measures should be involved. These are shown in Figure 6.

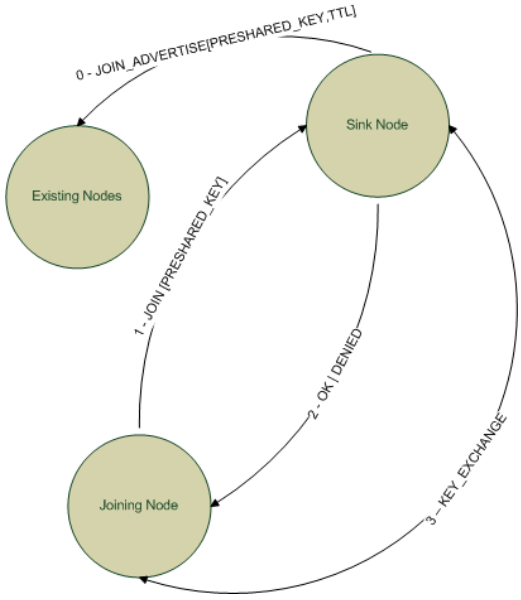

Figure 6 - Secure Join Mechanism

First, the sink node (connected to a computer) sends a secure layer 2 broadcast message advertising all the nodes in the network that a node with a given pre-shared key may join the referred network. This key as a Time To Live (TTL), meaning that for security purposes it will only be valid for a defined time according to the specific scenario.
Then, when the joining node wants to join the network, it uses a ROM burned pre-shared key to issue a join request (layer 2 communication with shared key security) to the sink node (through other nodes if multihop is used). If the join request is approved, the key exchange then starts so that the node receives the network layer 2 key and any specific layer 3 keys used to end-to-end communication.

3.6.2. Elliptic Curve Diffie-Hellman

Elliptic Curve Diffie-Hellman (EC-DH) is a cryptosystem which shares same ideology behind Diffie-Hellman but uses an elliptic curve as base. An elliptic curve is a plane curve which consists of the points that satisfy the equation

$$y^2 = x^3 + ax + b$$

along with a distinguished point at infinity, denoted ∞.

An example of a simple elliptic curve may be found in the next figure:

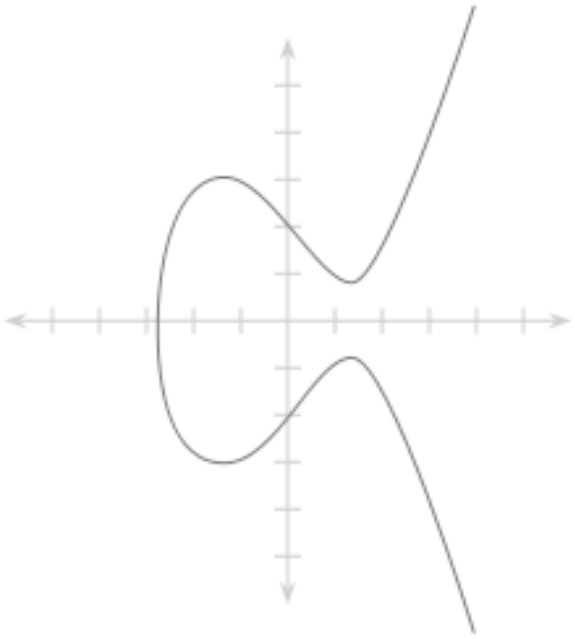

Figure 7 - Elliptic Curve Example [16]

As other asymmetric cryptography systems, it is based on the intractability of mathematical problems. In this case, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible. This was first suggested by Neal Koblitz[17] and Victor S. Miller[18] in 1985.
Considering this, Elliptic Curve Diffie-Hellman is considered more secure than the common Diffie-Hellman by NIST [19]. In fact, it requires a shorter key to deliver the same security performance as Diffie-Hellman. NIST also states that that EC-DH key should be twice the length of equivalent strength symmetric key algorithms as the following table demonstrates:

Symmetric Key Size (bits)	Diffie-Hellman Key Size (bits)	EC-DH Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Table 1 – NIST Recommended Key Sizes

But security is not the only attractive feature of elliptic curve cryptography. Elliptic curve cryptosystems also are more computationally efficient than the first generation public key systems such as RSA and Diffie-Hellman. Although elliptic curve arithmetic is slightly more complex per bit than either RSA or DH arithmetic, the added strength per bit compensates any extra computing time. The following table shows the ratio of DH computation versus EC-DH computation for each of the key sizes listed in table 1:

Symmetric Level (bits)	Ratio
80	3:1
112	6:1
128	10:1
192	32:1
256	64:1

Table 2 - Relative Computation Costs of DH and EC-DH

As far as key exchange is concerned, the algorithm is identical as already referred. It may be defined by the following steps:

- ✓ Two partners, Alice and Bob, agree on a elliptic curve, its underlying field and the base point.
- ✓ Alice and Bob generate a random value in the range $(1, r-1)$ where r is the order of the elliptic curve. This value will be the secret key of each of them.
- ✓ Then, they both calculate their public key by multiplying their secret keys times the chosen point on the curve.
- ✓ After that, they both exchange their public keys. After receiving the public key of the other partner, each of them multiply it with their secrete key. They both obtain the same number, which will be the key for symmetric cryptography.

However, this poses some problems for architectures with 8 or 16 bits because it keeps using large numbers. This is an issue that has to be solved and admitted, an can be overcome through the use of extensions that represent big numbers using 8 or 16 bits words and use carries to implement the arithmetic operations.

Even though EC-DH still uses big numbers and complex operations, the following table illustrates the differences between implementations of DH and EC-DH in TelosB motes. For comparison purposes, DH with 3072 bits and EC-DH with 256 bits have been used. The primes and curves are the ones suggested by NIST for testing purposes. To obtain statistically relevant data, 30 samples were collected at each node and the presented values are the average of those samples with a standard deviation of 5%. Time and energy values were measured using the mote's internal clock and the Energest (energy measuring) module of Contiki operating system. RAM and ROM values have been obtained with the tool msp430-size (included in the MSPGCC compiler) and represent accurate values measured only once.

Metric	**Diffie-Hellman**	**EC-DH**
ROM (bytes)	2902	3420
RAM (bytes)	458	635
Time on each node (seconds)	100,58	1,6
Power Consumption on each node (µW/h)	50274	802

Table 3 - DH versus EC-DH

Considering this results, EC-DH was chosen as the key distribution algorithm for the security model proposed in this thesis. Additionally, its measured performance was improved with a set of optimizations based on observation and study from the author of this dissertation:

- ✓ Assembly Coding: parts of the code were rewritten to Assembly in order to save cycles introduced by inefficiencies of the C compiler.
- ✓ Memory Usage Reduction: usage of RAM was decreased by eliminating unneeded variables and debug outputs.
- ✓ Optimize Arithmetic Operations: representation of big numbers with parts that were 0 have been removed from the operations.
- ✓ Pre-calculated values: in some cases where communication always happens with the same node (sink node), some values may be pre-calculated. This means that the agreement on the curve and the base point is established at programming time and do not have to be agreed and communicated at the key exchange process.

3.7. XOR Module

Some industry leading standards such as WirelessHart and Zigbee do not provide mechanisms to enable security at the physical layer of the communication stack. This makes them vulnerable to protocol violations and man in the middle attacks as explained before. In order to prevent most of these attacks, a module was defined at the proposed security model.
This module is named XOR as the main operation it uses is an exclusive OR, a very efficient computing operation. Essentially, it uses one byte of the layer 2 shared key and RX/TX counters to produce the number that will be used to XOR the bytes that are sent or received. As XOR is an operation that is reversible doing exactly the same, both

the sender and the receiver perform the same operation with data they both share. Without knowing the number and considering that it will change every time, the attacker will not decrypt the protocol that is being used. This will put some extra effort on breaking in attempts while using low computational resources and a small amount of memory ($N * N\ bytes$) where N is the number of nodes in the network/the number of nodes communicating.
The process that nodes use is the following:

- ✓ The sender (x) takes the number of messages that it has sent to the receiver (y) given by $N(x, y)$.
- ✓ It then computes every byte at the TX buffer (B), excluding the field with the number of bytes and a special field with the sender ID (at the beginning of the buffer), with the formula

  $$B\ XOR\ (KEY[N\ mod\ 16] +\ N\ mod\ 256)$$

  where B KEY is the layer 2 key (16 bytes) shared by every node.
- ✓ Then it sends the contents of the TX buffer to the other node.
- ✓ After the receiver gets the message, it takes the $N(x, y)$ where x is the sender and y itself (the receiver) and which represents the number of traded messages between the two nodes and computes the same XOR operation with the same formula explained above.
- ✓ Finally, the receiver has the translated message.

3.8. Watchdog

With dynamic security levels requirements and attack detection needs, a watchdog running at the security layer is a logical decision for a module addition.
Mainly, it has the following responsibilities:

- ✓ Keep track of keys at memory, which are stored at RAM for software implementations and at the radio transceiver buffers for hardware implementations. If there is an order to change keys from the Key Distribution Center (KDC) which is the sink node, the watchdog does that change and notifies the involved modules;
- ✓ Maintain pre-shared keys register and track their TTL to allow their expiration as programmed;
- ✓ Maintain a register of anomalous situations that may indicate attacks;
- ✓ Monitor resources such as battery to act accordingly;
- ✓ Share general information between the modules.

The following table illustrates the occurrences and the actions that the watchdog should as its default behavior:

Occurrence	Action
Battery Drops 5%	Lower the security level until battery is changed
Resources Occupation is High (> 90%)	Lower the security level for 10 minutes
KDC Orders Key Renewal	Notify involved modules to renew key and stop forwarding data while not renewed
Pre-shared Key of Other Node Expires	Notify MAC layer and symmetric cryptography module
Order to Change Security Level	Change security level
Abnormal Traffic	Notify the sink and wait for orders
High Number of Integrity Verification Fails	Notify the sink and wait for orders
Order to Ignore Compromised Node	Notify routing layer to re-route traffic

Table 4 – Watchdog Actions

These occurrences and their corresponding actions have been decided as the default behavior for the watchdog after tests and observation. In fact, the first two occurrences state specific values that could not be used without proper testing.
Regarding the battery, the 5% value was observed in tests that were used for autonomy validation and which are specified in the next chapter. The value is related with the boundaries of battery levels on which motes can operate.
In terms of resources occupation, the occurrence of more than 90% of resources in use and the action of lowering the security level for 10 minutes have been decided after testing a network with a high load, mostly because it was the combination that offered a better tradeoff between security not being too low while saving a good level of resources to accommodate peaks of high load.

3.9. Dynamic Security Levels

When developing any security system, a key point to focus on is the definition of security levels. With such definition it is possible to adjust the level of security according to the system's conditions or pre-defined policies.
For instance, a given network may not need confidentiality or it may not be a major concern, while for other it may be critical. Also, depending on the platform, there may be factors which become more important than security and cause the change for a lower security level to save resources dynamically.
Considering this, a set of basic security levels which can be changed dynamically by the Watchdog has been defined in table 5. However, the API allows the definition of custom levels so that they may be adjusted to specific scenarios.

Level	Key Distribution	Layer 1	Layer2	Layer3
Extreme	EC-DH 256 bits	XOR	AES-128-CCM	AES-128 (Full)
High	EC-DH 256 bits	XOR	AES-128-CCM	AES-128 (Auth Only)
Medium	EC-DH 256 bits	XOR	AES-128-CCM	-
Low	EC-DH 256 bits	XOR	AES-128-CTR	-
Very Low	EC-DH 256 bits	XOR	-	-

Table 5 – Default Security Levels

The table above represents the five security levels defined by default in the proposed security model. Key distribution is fundamental for the node to get key renewals, while layer 1 security with the XOR module is the most basic security that can be applied. Therefore, these two modules are the basis for all the default security levels.

However, the same does not apply to other modules. At layer 2, there are multiple combinations defined by IEEE 802.15.4. In this case, it only makes sense to use CCM (both integrity/authentication and confidentiality) or CTR (confidentiality), bearing in mind that the first uses twice the resources of the second and adds overhead (MIC) to the frame. Therefore, when optimizations are not possible at layer 3, adjust should be done to use CTR only. In alternative, CBC can also be configured if the focus of security is more on integrity than confidentiality.
Finally, at layer 3 there is also the possibility of using AES as encryption and integrity/authentication mechanism or solely as integrity/authentication algorithm. Even though AES-128 is referred in the table, for authentication/integrity the algorithm in use may be different if hardware AES-128 is not supported. In the case of this model, MD5 should be the mechanism in use.

4. Model Evaluation

4.1. Tests Specification

In order to validate the proposed model, a number of tests are defined. The first category of tests involves resources utilization, and aimed to evaluate the performance impact of the security model in industrial scenarios using a 3-2-1 (3 levels) tree topology and a TDMA MAC Layer with a simple application. The second category involved integration tests, and aimed to evaluate the compatibility, interoperability and portability of the model.
The first set of tests is aimed at the first category. Essentially, a number of metrics which are relevant for WSNs are defined and explained in the next section. These have been evaluated in the case study scenarios with its architecture with assurance of statistical representation by running the network for 24 hours and collecting huge samples that then were processed to obtain averages and standard deviations.
The last set of tests involved 2 motes: a TelosB [20] (16 bits architecture) running TinyOS and a Jennic JNB5139 [21] (32 bits architecture) running Contiki operating system. Both had the proposed security model implemented, and were set to communicate with 6LoWPAN sending the environment temperature every 3 seconds. A successful communication link and operation for 24 hours was defined as the objective to conclude that the security model integrates well at any architecture, with any protocol at most operating systems.

4.2. Evaluation Metrics

As already discussed, there are many obstacles to apply security to WSNs. These range from the limitation of resources to their wireless communication limitations. Therefore, a validation of any security model should pass at first level for the evaluation of metrics regarding these limitations.

4.2.1. Battery Life

Before the presentation of the evaluation studies, it is important to understand the consumption properties of the batteries used in the TelosB nodes. This knowledge will be useful to fully interpret the battery consumption evaluations.
The main restriction in the WSNs components, the nodes, it is the limited access to power supplies. Most nodes use simple AA batteries, which have a very particular discharge process.
A study performed over MSP430 hardware [22] concluded that in the beginning of the lifetime, the battery energy decreases slightly faster than past its middle. Once in the middle, the battery consumption is constant. The next figure shows the curve of the battery discharging achieved by Kramer et al:

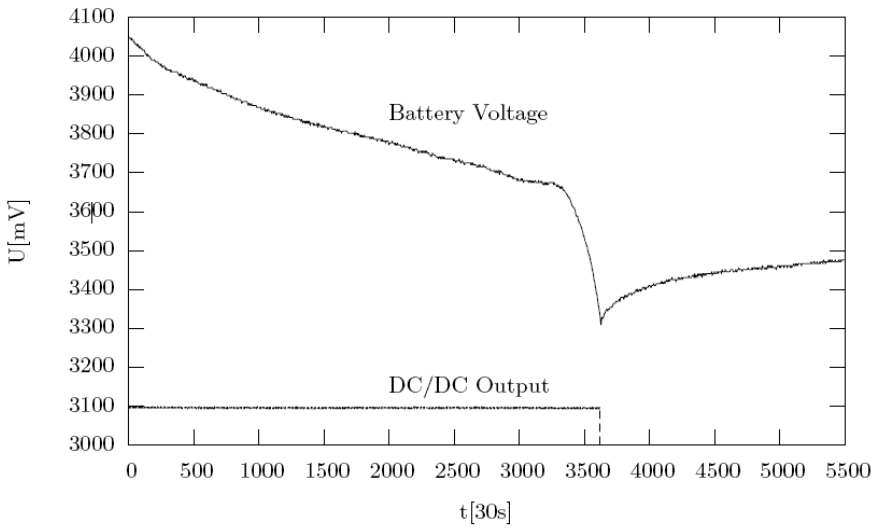

Figure 8 - Battery Consumption Example [22]

In real scenarios, these characteristics are common and not only when the battery is new. Every time the node is turned on, the battery requires a fixed time to become stable, achieving its real value. In the study it takes about 250x30 seconds for the battery to stabilize.
For instance, the battery consumptions in TelosB motes (according to its datasheet) are reported in samples defined by the follow expression:

$$Vbat = \frac{Vref \ x \ ADC\_FS}{ADC\_count}$$

Where: Vbat = Battery Voltage, Vref = Internal Voltage reference = 1233mV, ADC_FS=1024, and ADC_count is equal to measured value.
As there is a conversion from an analog value to a digital value through an analog to digital converter (ADC), the sampling rate causes some loss of information.
If the measured value (ADC_count) was 475 (decimal units converted from the hexadecimal read value), the Vbat would be equal to 2636,5305. The next measured value would be 476, which means a Vbat equal to 2630,9916. Hence, the reported battery value never decreases in one by one unit. Instead, it decreases in samples due to the TelosB conversion formula (for instance 6 in 6 mV). This characteristic is very important to understand the evaluations presented ahead in this thesis.

4.2.2. Performance and Resources Impact

In order to evaluate performance and resources impact over a network setup in a critical environment such as the one described in the next section, a number of metrics have been defined. The first one was latency.
On specific scenarios, it is critical that the delay between a given event and its acknowledgement at the destination is deterministic and below a certain threshold. Therefore, any security mechanism that runs continuously cannot have a great impact on the delay by taking too much time to run. To measure the end-to-end latency, clock synchronization was achieved through a specific module in Contiki operating system (nodes set their time based on messages from the master/sink node) and a timestamp at each packet provided the value to obtain the delay time. It was obtained with a precision of 1/32768, which corresponds to the maximum frequency of the hardware clock (32 MhZ) of the chosen platform.

The other two metrics were related with resources utilization. As multi-threading is not used in the test scenario, CPU was not considered relevant. However, even in other scenarios, while running security algorithms there should not be more resource intensive computation occurring and CPU would not be a problem. Hence, for this evaluation study the only metrics considered were the utilization of RAM and ROM. ROM is easy to evaluate, as the size of the binary that is programmed to the mote corresponds to its utilization. However, RAM utilization measurement was not immediate. To obtain its value, one needs to know that typically memory allocation is not dynamic. As it is static, there is a small tool included with MSPGCC compiler that gives the value of bytes allocated at RAM (msp430-size). With this tool, accurate results have been obtained.

4.3. Tests Results

The results for the defined tests, both on industrial scenarios and integration scenarios, are presented and described below.

4.3.1. Performance, Resources and Autonomy

In the following plots, the results in the integration scenarios are presented.
The first results are related with memory consumption, namely ROM (Fig. 9) and RAM (Fig. 10). The plots for both metrics show the consumption when the mote supports hardware cryptography (first bar) and when it does not (second bar). On each bar, it is possible to observe the base utilization (Contiki OS and applicational software) together with the sum of each module of the security model.

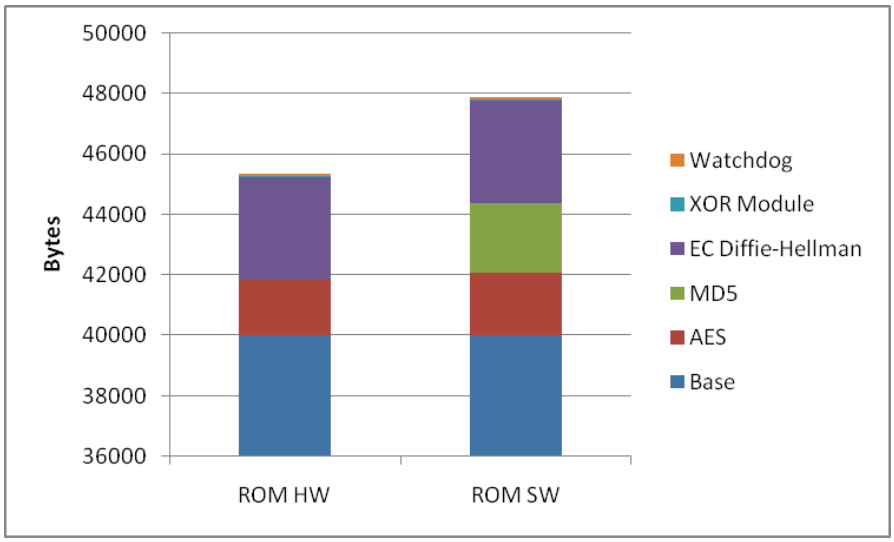

Figure 9 - Security ROM Usage

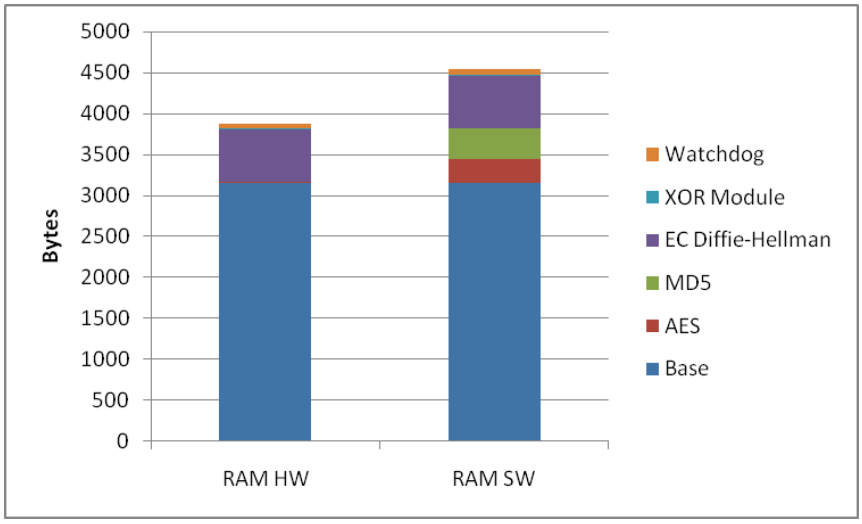

Figure 10 - Security RAM Usage

From these 2 plots, it is possible to conclude that the security model application is feasible given the limitations of the platform (48KB of ROM and 10KB of RAM). However, although RAM consumption is very low for both bars, when software cryptography is needed the amount of ROM for expansion is very small. This may pose a problem due to the high base utilization of ROM for the industrial scenario application, but as the chosen platform for this case is TelosB (with hardware cryptography support) there is enough space to insert more code.
Also, it is possible to validate the model requirement of RAM and ROM utilization. As the defined requirements were 2KB of RAM and 8KB of ROM at most, here we may conclude that even with all implementations at software level these resources never get past those values.

The second results demonstrate the end-to-end latency evaluation by tree levels. This is due to the tree topology, which normally affects the delay as multihop is needed. Level 1 is considered the tree branch below the root (sink node), level 2 the middle branch and level 3 the branch which includes the leaf nodes. For each of these levels, end-to-end latency (node to sink) was measured without security, with hardware based cryptography and also with software based cryptography. This evaluation can be observed in Figure 11.

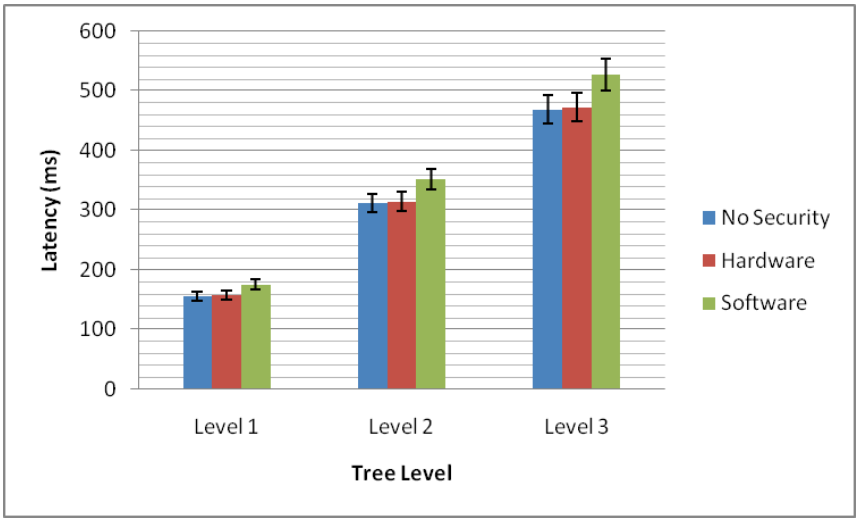

Figure 11 - Security Latency Impact

From this plot, we may conclude that latency is highly affected by the level at which the node is placed. In fact, the relation is roughly N x Level 1 Latency where N is the node's level. Additionally, the impact of the security model when hardware cryptography is used is very low and inside the margin of the standard deviation which produces a irrelevant difference. However, the same does not apply to software only based cryptography. In this case, the value is at most about 50 milliseconds above the non-secured environment. Even so, it is considered a good value as the requirements of the specific scenario state that everything below 1 second is inside the margin and the worst obtained value (~525 milliseconds) plus the standard deviation does not get past 560 milliseconds.

Finally, the last results represent the battery level (autonomy) of the network with an average from all the tree levels evaluated by the type of security applied. It was evaluated for the network running with no security, with hardware based cryptography and with software only based cryptography. Also, for both hardware and software cryptography implementations, tests were conducted with and without the application of security levels through the monitoring done by the watchdog. The results are presented in the next figure:

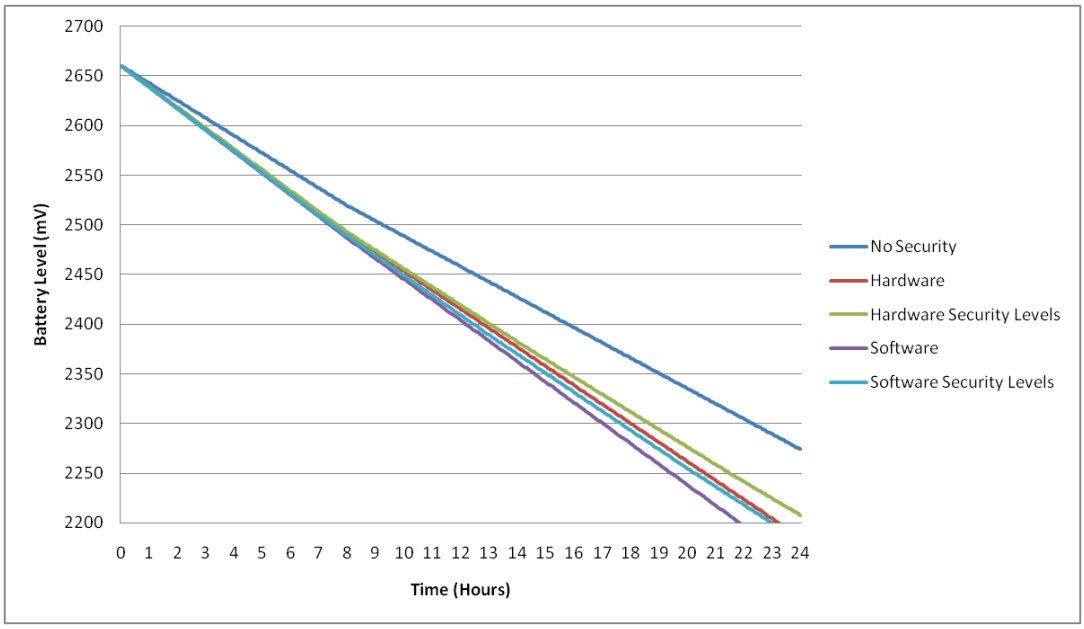

Figure 12 - Security Battery Discharge

From the lines at the above plot, the theory analyzed at sub-section 4.2.1 regarding the battery discharge rate being higher when the battery is full is proved. In fact, after 8 hours of running time the discharge rate began to stabilize at a lower value. Additionally, the decrease of 6 mV at each time is also observable as there are some small (at this scale) steps along the lines.

Other conclusion from this evaluation was that after the battery (two AA batteries) decreases below 2200 mV, the node stops working properly. This was later confirmed by the TelosB datasheet that states that below around 2100 mV the ADCs and other components stop working properly. Therefore, the results presented here only considered the interval until the value gets below 2200mV.

Regarding the results themselves, it is possible to observe that without security the autonomy of the nodes gets easily past the 24 hours (study duration). However, there is not much difference when comparing to hardware-based cryptography, especially when the security levels are dynamically adjusted to fit the remaining energy. As for software-based cryptography, the results show that the autonomy does not go after the 23 hours. However, this is still a very good value and shows a low impact (about 2 hours a day) for a security model with such a level of security relying solely on software implementations.

4.3.2. Contiki – TinyOS Integration

As explained on sub-section 4.1, integration tests between Contiki OS and TinyOS operating systems were conducted. Also, different architectures and different protocols implementation were integrated in order to ensure that the security model proposed in this thesis is in fact interoperable.
After setting up the scenario and turning on the motes, the first problem came to light. There was no communication because the duty cycle (radio ON/radio OFF) schemes of the MAC Layers in Contiki and TinyOS were different. The easiest solution was to use a NullMAC implementation (radio always ON), which led to communication occuring at layer 1. However, MAC Layers were still no communicating. This was a much harder to solve issue, and only after digging into hundreds of lines of code the problem became clear. There are differences in the implementation of IEEE 802.15.4 of both Contiki and TinyOS.
The most relevant one, and the only one that is important enough to mention, was the size of the MAC addresses. Contiki uses extended IEEE 802.15.4 addresses (64 bits) as TinyOS uses short IEEE 802.15.4 addresses. This is configurable and an option at the standard, which is defined at the MAC Frame.

After understanding this issue, the solution seemed to be easy. Change one of the operating systems' configurations to use the same type address of the other. However, the solution was not so evident because upper layers also depend on these addresses. In fact, 6LoWPAN uses the 802.15.4 address to derivate the suffix for the IPv6 address and the algorithm is once again different in both TinyOS (BLIP) and Contiki (SICSLoWPAN). As the author of this thesis is more familiar with the implementation in Contiki, it was decided to change the address size in Contiki and adjust the algorithm at SICSLoWPAN that derivates the suffix.
Next, the communication stacks were finally communicating but packets where not arriving at the application layer. It was one more difference of the 6LoWPAN implementation, now regarding the header compression algorithms. After adjusting both implementations to use the most basic header compression algorithm (different types are available), data started arriving at the application of both motes without problems. This ran successfully for 24 hours with the security model on at the highest level of security with both hardware and software based cryptography.

5. Conclusions and Future Work

A fully functional security model with security, performance, interoperability and reliability was obtained. This is supported by the tests included in section 4, which clearly prove that the proposed security model is valid and easily integrated. This last statement is in fact well proven, as the integration issues to test compatibility, portability and interoperability were all related with protocols and underlying operating systems not following the standards or using different options of implementation.

Although the model proposed and validated in this work is an improvement over other solutions and due to its open source characteristic may be attractive to the open source community represented by Contiki and TinyOS operating systems, there is still much room for improvements. These improvements can be summarized by the following topics:

- ✓ Improve attack detection and dynamic security measures.
- ✓ Make adjustments or find new algorithms to enhance the performance of software cryptographic implementations.
- ✓ Optimize key distribution and key renewal procedures.
- ✓ Study and evaluate better layer 1 security mechanisms with equivalent performance.
- ✓ Study and test authentication mechanisms to join the network for the first time and to improve general authentication. Radio patterns, which are like fingerprints of the radio transceivers, could be an option.

Additionally, more improvements are always possible and technology is evolving at a rhythm that hungrily demands new security systems and mechanisms as privacy and security concerns arise. Hence, the subject of this thesis and the results obtained here can be used as a point of start for future work of other students and researchers.

References

[1] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LRWPANs), IEEE Std. 802.15.4, Sep. 2006. [Online]. Available: http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf

[2] Sastry, N. and Wagner, D. Security Considerations for IEEE 802.15.4 Networks. In WiSe '04: Proceedings of the 3rd ACM workshop on Wireless security, pages 32–42, New York, NY, USA, 2004. ACM Press.

[3] D. Gascón. (2011, Jan.) IEEE 802.15.4 and ZigBee Security. [Online]. Available: http://www.sensor-networks.org/index.php?page=0903503549

[4] ZigBee Alliance. (2011, Jan.) ZigBee. [Online]. Available: http://www.zigbee.org

[5] J. Cobb, E. Rotvold, J. Potter. (2010) 'WirelessHART Security Overview'. [Online]. Available: http://www.hartcomm.org/protocol/training/resources/wiHART_resources/Security_Overview_LIT114.pdf

[6] Daemen J, Rijmen V. The Design of Rijndael: AES - The Advanced Encryption Standard. First edn., Springer, 2002.

[7] National Institute of Standards and Technology. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197 Nov 2001. URL http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[8] Law YW, Doumen J, Hartel P. Survey and benchmark of block ciphers for Wireless Sensor Networks. ACM Transactions on Sensor Networks TOSN Feb 2006; 2(1):65–93.

[9] Karlof C, Sastry N, Wagner D. TinySec: A link layer security architecture for Wireless Sensor Networks. Proceeding of the 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, MD, 2004; 162–175.

[10] Rivest R. The MD5 Message-Digest Algorithm. RFC 1321 Apr 1992. URL http://tools.ietf.org/html/rfc1321.

[11] National Institute of Standards and Technology. Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180-2 Aug 2002. URL http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

[12] Granjal J, Silva R, Monteiro E, Silva JS, Boavida F. Why is IPSec a viable option for Wireless Sensor Networks. Proceedings of the Fifth IEEE International Conference on Mobile Ad-hoc and Sensor Systems, 2008; 802–807.

[13] Chipcon. CC2420 Datasheet Jun 2004. [Online]. Available: http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf.

[14] Didla S., Ault A, Bagchi S. Optimizing AES for embedded devices and Wireless Sensor Networks. Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities, 4, 2008; 10.

[15] Gladman, B. "Optimized AES Implementation". Fev. 2012 [Online]. Available: http://www.gladman.me.uk/

[16] Wikimedia. "Simple Elliptic Curve". Apr. 2012 [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/d/da/Elliptic_curve_simple.svg

[17] Koblitz, N. (1987). "Elliptic curve cryptosystems". Mathematics of Computation 48 (177): 203–209. JSTOR 2007884.

[18] Miller, V. (1985). "Use of elliptic curves in cryptography". CRYPTO 85: 417–426. DOI:10.1007/3-540-39799-X_31.

[19] NSA. "The Case for Elliptic Curve Cryptography". Apr. 2012 [Online]. Available: http://www.nsa.gov/business/programs/elliptic_curve.shtml

[20] C. Technology®. (2011, Jan.) TelosB™ Datasheet. [Online]. Available: http://www.willow.co.uk/TelosB Datasheet.pdf

[21] NXP Semiconductors. (2011, Jan.) Jennic JN5139. [Online]. Available: http://www.jennic.com/products/wireless_microcontrollers/jn5139

[22] M. Kramer and A. Gerldy, "Energy Measurements for MicaZ Node," Fachgespräch "Dahtlose Sensornetze", GI/ITG KuVS, 2006