Radboud Universiteit Nijmegen

The Kerckhoffs Institute
Institute for Computing and Information Sciences

Reverse Engineering WirelessHART Hardware

Master Thesis

Eduardo Pablo Novella Lorente

Student Number: *s*4253043

Supervisors:
Dr. ir. Harald Vranken (RU)
Daniël Niggebrugge (Fox-IT)

Public Version

Nijmegen and Delft, (The Netherlands)
August 2015

Abstract

A Wireless Sensor Network (WSN) is a group of wireless nodes that use radios to monitor and record environmental and physical conditions. Wireless Sensor Networks (WSNs) are an emerging trend to resolve plenty of real world issues and are becoming an interesting and cutting-edge topic for researchers to solve many challenging situations. These wireless sensors are responsible for monitoring data from applications such as medicine, army, smart-cities, animal monitoring... All these communications are crucial and need to remain secure during all the end-to-end communications in order to avoid the interception of adversaries. Although these wireless nodes use very secure cryptographic algorithms, the physical security has to be taken into account as well.

This thesis is principally focused on a physical attack, concretely the "*node capture*" attack on the industrial automation control standard named *WirelessHART*. Low-cost physical attacks have been carried out on a couple of *WirelessHART* sensor nodes. The "*node capture*"attack relies on full access control over the sensor node through direct physical access. Wireless nodes contain micro-controllers, which have several security measures to stop the reading of their internal content. Such protections are not enabled by default and they need to be analysed carefully before their deployment. Between the low-cost attacks, the *JTAG* interface is one of the first backdoors to the system which allows us to interact with the micro-controller for debugging their system, memory and I/O ports. This thesis addresses several attempts to read out memory contents from various wireless nodes.

The main goal of this paper is to attempt to extract the *JoinKey* from a couple of *WirelessHART* nodes after their "*node capture*" attack. If an attacker was able to extract this key from a sensor node, he/she would be capable to freely join to the network. Allowing him/her to interact with the rest of nodes in a legitimate way and manipulate information or commit '*sinkhole*'-'*wormhole*' attacks in order to provoke a denial of service in the network. Finally, this thesis concludes with a successful key retrieval for a development wireless node and not successful for a real *WirelessHART* device.

Release Notes

The present document is the ***Public version*** of a research project carried out as part of an internal Fox-IT project. This master thesis contains the foundations of *Reverse Engineering WirelessHART Hardware* project, omitting specific information (e.g. brands, manufacturers, real cryptographic AES keys, etc). For further information you may contact Fox-IT.

Fox-IT B.V.
Olof Palmestraat 6
P.O. box 638
2600 AP Delft
The Netherlands

Phone: +31 (0)15 284 7999
Fax: +31 (0)15 284 7990
E-mail: info@fox-it.com
Internet: http://www.fox-it.com

Preface

This Master's thesis is the result of my graduation project executed at Fox-IT in The Netherlands. The graduation project is part of my Master in Computer Science and Engineering - *Information Security Technology* at the Radboud University Nijmegen. This security-focused master program is coordinated by the Kerckhoffs Institute, a collaboration program between the University of Twente, Radboud University Nijmegen and the Eindhoven University of Technology.

I would like to express my gratitude to everyone who helped me carry out my MSc thesis. Concretely to my supervisors, Dr. ir. Harald Vranken (Radboud University Nijmegen) and Daniël Niggebrugge (Fox-IT), who gave me feedback and the correct guidelines to keep going further in my research. Additionally, some workmates who cheered me up during the long journey until my graduation. Furthermore, I would like to name Max Duijsens, a TU/e student who was attempting to fuzz the WirelessHART protocol, with whom I clarified some details of this thesis.

Of course, I would like to thank my family and best friends, especially my mother and older sister. Without their trust and support, I would have never arrived here. They played an important role in my career, as well as my good friends who I have not been able to see since I had moved out to The Netherlands.

Finally, I am very thankful the three responsible organizations of the Kerckhoffs Institute programme and especially to my supervisors and other teachers such as Lejla Batina, Tanja Lange, Erik Poll, Joeri de Ruiter, Peter Schwabe and Roel Verdult. First, I have no words to describe my experience at Radboud. It was an amazing period of time in my life where I never stopped learning and growing up, especially during my Erasmus year (2012) and MSc programme at Radboud University Nijmegen (2013-2015).

Eduardo Pablo Novella Lorente

April-July, 2015

Contents

Contents vi

List of Figures ix

List of Tables xi

Abbreviations xii

1 Introduction 1
- 1.1 Introduction . . . 1
- 1.2 Goals . . . 2
- 1.3 Research Method . . . 3
- 1.4 Outline . . . 3

2 Background 5
- 2.1 Wireless Sensor Networks Applications . . . 5
  - 2.1.1 Smart-houses . . . 5
  - 2.1.2 Smart-cities . . . 6
  - 2.1.3 Habitat monitoring . . . 9
  - 2.1.4 River monitoring . . . 10
  - 2.1.5 Smart-Metering . . . 10
  - 2.1.6 Warfare: Tracking enemies and tactical techniques . . . 11
  - 2.1.7 Natural disasters . . . 12
  - 2.1.8 Agriculture and animals tracking . . . 12
- 2.2 Sensor node Architecture . . . 13
  - 2.2.1 Microcontroller (MCU) . . . 13
  - 2.2.2 Radio . . . 14
  - 2.2.3 External Memory . . . 16
  - 2.2.4 Power source . . . 16
- 2.3 Highway Addressable Remote Transducer (HART) . . . 16
  - 2.3.1 HART Protocol . . . 16
  - 2.3.2 Modulation: Audio Frequency Shift Keying (AFSK) . . . 17
  - 2.3.3 Communication Modes . . . 17
  - 2.3.4 Network Configurations . . . 18
  - 2.3.5 HART communication layers . . . 19
  - 2.3.6 Packet Structure . . . 19
  - 2.3.7 HART commands . . . 20
  - 2.3.8 HART-IP . . . 20
- 2.4 IEEE 802.15.4 standard . . . 22
  - 2.4.1 Physical layer (PHY) and Media Access Control (MAC) layers . . . 22
  - 2.4.2 Comparison of IEEE 802.15.4 standard and other wireless technologies . . . 23
  - 2.4.3 IEEE 802.15.4 Device Classes . . . 24

2.4.4 IEEE 802.15.4 Network Topologies . . . 24
2.4.5 IEEE 802.15.4 Packet Structure . . . 25
2.5 WirelessHART . . . 26
2.5.1 Main Characteristics . . . 26
2.5.2 WirelessHART Components . . . 27
2.5.3 Wireless Highway Addressable Remote Transducer Protocol (WirelessHART) Communication Layers . . . 28
2.6 Cryptography in WirelessHART . . . 33
2.6.1 Outline . . . 33
2.6.2 Advanced Encryption Standard (AES) . . . 33
2.6.3 AES Counter (CTR) . . . 34
2.6.4 AES Cipher Block Chaining (CBC) . . . 34
2.6.5 AES Cipher Block Chaining Message Authentication Code (CBC-MAC) . . . 35
2.6.6 AES Counter with CBC-MAC (CCM)* . . . 35
2.7 Key Management in WirelessHART . . . 36
2.7.1 Cryptographic Keys in WirelessHART . . . 36
2.7.2 Joining Process . . . 37
2.8 Others 802.15.4 Wireless Protocols . . . 45
2.8.1 ZigBee . . . 45
2.8.2 ISA100 . . . 45

3 Hardware Security **47**
3.1 Overview . . . 47
3.2 Hardware protocols . . . 47
3.2.1 Serial Peripheral Interface (SPI) . . . 47
3.2.2 JTAG . . . 48
3.2.3 UART . . . 50
3.3 Hardware Security . . . 50
3.3.1 Hardware protections: Fuses and lock bits . . . 51
3.3.2 Bootstrap Loader (BSL) . . . 52
3.4 Physical Attacks on Wireless Sensor Networks . . . 53
3.4.1 Consequences and possible attacks . . . 54

4 Tools **57**
4.1 Outline . . . 57
4.2 Bus Pirate v.3.6 . . . 57
4.3 GoodFET v.42 . . . 58
4.4 TL866A USB Universal Minipro Programmer . . . 58
4.5 AVR Dragon JTAG programmer . . . 59
4.6 ARM Segger J-Link JTAG programmer . . . 59
4.7 Open On-Chip Debugger . . . 59
4.8 Flashrom . . . 60
4.9 IDA Pro and Hex-Rays . . . 60

5 Targets **61**
5.1 Outline . . . 61
5.2 Linear SmartMesh WirelessHART Starter Kit DC9007 . . . 61
5.2.1 Hardware Components: Mote DC9003A-C . . . 62
5.2.2 Setting up a WirelessHART network . . . 63
5.2.3 Identifying programming interfaces in the Mote DC9003A-C . . . 63
5.2.4 UART . . . 64
5.2.5 JTAG . . . 64
5.2.6 SPI and the Hardware Lock Key . . . 66
5.2.7 Firmware and JoinKey extraction . . . 66

5.3 Linear Access Point Manager LTP5903CEN-WHR 69
5.3.1 Firmware extraction 69
5.3.2 Credentials Found 69
5.4 A WirelessHART `unknown-mote` 70
5.4.1 Hardware Components: `unknown-mote` 70
5.5 Microcontroller TI MSP430 71
5.6 AVR Microcontroller 72
5.7 The BGA chip: DN2510 74

6 Conclusion **77**
6.1 Conclusions 77
6.2 Further Work 78

Bibliography **81**

List of Figures

2.1 Physical specifications of a Telos Revision B. 2004 7
2.2 General architecture for wireless sensor nodes 14
2.3 Comparison between WirelessHART Radio Modules and their main specifications. 15
2.4 AFSK. Simultaneous Analogue and Digital Communication 17
2.5 HART Master-Slave communication between a Distributed Control System (DCS) and field device. 18
2.6 HART two masters communication between a Personal Computer (PC), handheld terminal and field device. 18
2.7 Network topologies in 802.15.4 25
2.8 Packet structure in PHY and MAC layers for the IEEE 802.15.4 standard. 26
2.9 WirelessHART Mesh Network Architecture 28
2.10 Packet structure in HART. 29
2.11 Packet structure in WirelessHART. 30
2.12 Dissection of a WirelessHART packet at different layers. 31
2.13 Security features comparison between HART and WirelessHART. ([1] improved) . 32
2.14 AES-CTR encryption mode 34
2.15 AES-CBC encryption mode 35
2.16 AES-CBC-MAC authentication mode. Calculation of the Message Integrity Code (MIC) or MAC. 36

3.1 Circular buffer using two shift registers in a SPI communication. 47
3.2 Joint Test Action Group (JTAG) architecture. Boundary scan and Test Access Port (TAP) controller. 48

4.1 The open source hardware hacker tool 'Bus Pirate' v3.6 used for interacting with chips. 57
4.2 The open source JTAG programmer 'GoodFET' used for debugging several wireless radio chips. 58
4.3 The Atmel AVR Dragon programmer. 59

5.1 Dust Networks/Linear SmartMesh WirelessHART Starter Kit 62
5.2 Linear Mote DC9003A-C architecture and its ARM Cortex M3 Eterna LTC5800. . 63
5.3 Linear WirelessHART Kit set up. 64
5.4 Setting up a JoinKey through the Linear Mote Eterna API over the Universal Asynchronous Receiver/Transmitter (UART) interface. 64
5.5 JTAG'ing the Eterna Mote-on-chip with OpenOCD and J-link ARM JTAG USB. . 65
5.6 Connecting to the TCP socket and attempting to dump the flash on the Eterna Mote-on-chip. 65
5.7 Dumping the mote's memory content through SPI using Linear tools. 66
5.8 Dumping the mote's memory content through SPI using a Bus Pirate v3.6 67
5.9 Linear Mote Eterna firmware extraction by using Flashrom and Bus Pirate v3.6 . . 67
5.10 Memory dump of a Linear Mote Eterna and several Join Keys used. 68

5.11 Firmware extraction of the Linear Manager LTP5903CEN-WHR. 69
5.12 XML-RPC users and passwords not documented. 70
5.13 TI MSP430 MicroController Unit (MCU) discovered by using a microscope. 71
5.14 Hooking up cables in the JTAG pins in TI MSP430 MCU. 72
5.15 JTAG'ing the TI MSP430 MCU with the GoodFET. 72
5.16 Trying to JTAG the AVR chip with an AVR Dragon. 73
5.17 The internals of the Linear Access Point (AP) Manager and its M2510 highlighted with red colour. 74
5.18 (1) The Radio chip DN2510 in the unknown-target and possible pins for Serial Peripheral Interface (SPI). (2) The Radio chip in the Linear AP Manager which is readable through a socket. 75
5.19 Dumping out the M2510 Ball Grid Array (BGA) memory through a socket and the Bus Pirate v3.6. 75
5.20 Dumping the Linear DN2510 BGA memory by using a BusPirate v3.6. 76

List of Tables

2.1 HART protocol and OSI model. 19
2.2 Structure of a HART packet. 19
2.3 Dissection of a HART packet. 20
2.4 Examples of HART Universal commands 21
2.5 Examples of HART Common Practice commands 22
2.6 Examples of HART Device Specific commands 22
2.7 Comparison between IEEE 802.15.4 PHY layer according to geographical regions. . 23
2.8 Comparison between IEEE 802.15.4 and the most well-known wireless technologies. 23

3.1 SPI logical signals. 48
3.2 JTAG logical signals. 50
3.3 UART logical signals. 50

Acronynms

6LoWPAN IPv6 over Low power Wireless Personal Area Networks

ACL Access Control List

AES Advanced Encryption Standard

AFSK Audio Frequency Shift Keying

AP Access Point

API Application Programming Interface

BGA Ball Grid Array

BGWK Broadcast Gateway Key

BNMK Broadcast Network Manager Key

bps bits per second

BPSK Binary Phase Shift Keying

BSDL Boundary Scan Description Language

BSL BootStrap Loader

BSR Boundary Scan Register

CBC Cipher Block Chaining

CBC-MAC Cipher Block Chaining Message Authentication Code

CCM Counter with CBC-MAC

CFB Cipher Feed Back

CMAC Cipher-based MAC

CPU Central Processing Unit

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

CTR Counter

DCS Distributed Control System

DES Data Encryption Standard

DLL Data Link Layer

DLPDU Data Link Layer (DLL) Protocol Data Unit

DSSS Direct Sequence Spread Spectrum

ECB Electronic Code Book

ECC Elliptic Curve Cryptography

FET Flash Emulation Tool

FFD Full Function Device

GPIO General Purpose Input Output

GPS Global Positioning System

GTS Guarantee Time Slots

GW Gateway

HART Highway Addressable Remote Transducer

HK Hand-held Key

HVPP High Voltage Parallel Programming

HVSP High Voltage Serial Programming

I2C Inter-Integrated Circuit

IC Integrated Circuit

ICSP In Circuit Serial Programming

IDA Interactive Disassembler

IEEE Institute for Electrical and Electronics Engineers

IoT Internet Of Things

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISA100.11a International Society of Automation 100.11a

ISM Industrial, Scientific and Medical

IV Initialization Vector

IVT Interrupt Vector Table

JK Join Key

JTAG Joint Test Action Group

LR-WPANs Low Rate Wireless Personal Area Networks

MAC Media Access Control

MCU MicroController Unit

MFR MAC Footer

MHR MAC Header

MIC Message Integrity Code

MiWi Microchip Wireless

MMC MultiMedia Card

MSDU MAC Service Data Unit

NDA Non Disclosure Agreement

NIST National Institute of Standards and Technology

NK Network Key

NM Network Manager

NPDU Network Protocol Data Unit

OCD On-Chip Debugging

OFB Output Feed Back

O-QPSK Offset Quadrature Phase Shift Keying

OS Operating System

PAN Personal Area Network

PCB Printed Circuit Board

PC Personal Computer

PDI Program and Debug Interface

PHR PHY Header

PHY Physical layer

PKI Public Key Infrastructure

PLC Programmable Logic Controller

PPDU PHY Protocol Data Unit

PPDU Physical (PHY) Protocol Data Unit

PP Parallel Programming

PRNG Pseudo Random Number Generator

PSDU PHY Service Data Unit

PSK Pre-Shared Key

PV Process Variable

RAM Random Access Memory

RFD Reduced Function Device

RFID Radio Frequency Identification

RF Radio Frequency

ROP Return Oriented Programming

SCA Side Channel Attack

SHM Structural Health Monitoring

SHR Synchronization Header

SM Secure Manager

SM Security Manager

SoC System-on-Chip

SPI Serial Peripheral Interface

TAP Test Access Port

TDI Test Data In

TDMA Time Division Multiple Access

TDO Test Data Out

TI Texas Instruments

UART Universal Asynchronous Receiver/Transmitter

UDP User Datagram Protocol

UGWK Unicast Gateway Key

UNMK Unicast Network Manager Key

WAPMS Wireless Sensor Network Air Pollution Monitoring System

WiFi Wi-Fi

WirelessHART Wireless Highway Addressable Remote Transducer Protocol

WKK Well Known Key

WSNs Wireless Sensor Networks

WSN Wireless Sensor Network

XOR eXclusive OR

Chapter 1

Introduction

1.1 Introduction

A Wireless Sensor Network (WSN) consists of a high number of autonomous sensor nodes and one or a few gateway nodes. Sensor nodes are called *Motes* due to their small size. These motes collect environmental information such as temperature, pressure, humidity, light conditions, movement and natural disasters using attached sensors. This information travels wirelessly from a mote through the network until it reaches a gateway. During the forwarding, different routes might be taken depending on the availability of the sensor nodes routing tables. The gateway node is normally connected to a computer from a different network in order to gather all the data sent from the WSN. The wireless sensor networks are a combination of distributed sensing, computing and communication [2]. These Wireless Sensor Networks (WSNs) are ad-hoc networks, where each sensor node participates in routing by forwarding information to other sensor nodes using the nearest neighbouring communication. Without the use of a pre-defined infrastructure until the gateway is reached. These gateways might be bridged with other gateways either to combine networks such as the Internet, or to extend the data transmission to a location that sensor nodes cannot reach.

The application areas diversify a lot due to the many different application environments for the use of such WSNs. Nevertheless, we can discuss some of the basic requisites which are common in many areas. These sensor nodes are normally self-powered by batteries or energy harvesting systems and they remain unattended for years in the wild. For this reason, the hardware devices must be as cheap as possible because sooner or later they will be disposed. WSNs can be constituted of hundreds or even thousands of sensor nodes spread out over many places. The battery degradation requires a dynamic voltage scaling during its lifetime to provide trustworthy information. Since the lifetime is limited because of batteries, the sensor nodes use MicroController Unit (MCU)s with very low power consumption and a vertiginous wake-up time. In spite of the fact that sensor nodes have resource constraints, they should still be tamper resistant. The MCUs sporadically have high computational load when they are processing distributed data, performing data aggregation or cryptographic operations like encryption, signing or decryption. Although the sensor nodes are using strong cryptographic protocols, they are exposed to hazardous environments where these nodes can be stolen by an adversary. Sensor nodes should be physically well-protected to avoid as many physical attacks as possible.

The use of sensor nodes is very well addressed in the industrial environment through the Highway Addressable Remote Transducer (HART) protocol. In such environments, many processes must be automated and monitored. An industrial environment is comprised of many field devices measuring different process values and subsequently transmitting them to the host application. The HART protocol has been used for a long time to interact in wired networks and it remains

very reliable for many operations. However, the industrial community noticed the need to reduce costs, strengthen the security and also to decrease the problems which had been arising due to the hazardous locations where the field devices were located.

Reaching enormous oil or gas pumps at remote locations might be a really intensive and time-consuming task for industrial workers. A solution is the Wireless Highway Addressable Remote Transducer Protocol (WirelessHART) standard. This standard inherits all the properties of the HART protocol while going wireless. Although many problems seem to have been resolved, going wireless does provoke some side effects. This thesis is addressing these problems and will offer security countermeasures and guidelines to properly protect MCUs. The HART protocol has neither authentication nor confidentiality at the different layers of the protocol. Contrary to the HART protocol, WirelessHART provides both data integrity and confidentiality for the protocol layers in the wireless part of the network. One of the most attractive ideas was to conserve the backwards compatibility, allowing wired and wireless devices to work together in order to create sensor networks.

For a long time, wireless technologies have been a real target for adversaries due to the easiness of intercepting traffic and attacking without being seen as well as some weaknesses in their security protocols. WSNs are also a big target due to the importance of the information they hold. For many wireless applications, the key management relies on a Pre-Shared Key (PSK) which must be established before communicating with each other. This PSK is known in WirelessHART as the Join Key (JK). The JK is the indispensable requirement for becoming a member in the WSN and is considered to be thoroughly hidden from external devices and even different wireless networks. Owning the JK allows attackers to be authenticated into the wireless network and receive the rest of the session keys, allowing them to interact within the network. When a JK is compromised, the security is overcome and plenty of attacks can easily occur. To prevent this, WSNs have developed new state-of-the-art algorithms to detect possible node capture attacks and measure timing delays in order to predict when a node could be attacked. A successful node attack would be an attack which is undetectable to its neighbours and ensures that the node remains alive, transmitting and routing traffic as it normally does. If this is achieved, then the entire network is considered to be tainted.

1.2 Goals

In order to achieve some of following goals, a *black-box reverse engineering* research has been carried out. Other aims have been accomplished by merging many papers and attempting to clarify some doubts regarding the WirelessHART joining process. The principal objectives of the present thesis are depicted as follows:

- Address low-cost physical attacks on a couple of wireless nodes. This thesis focuses on debugging and programming interfaces (Joint Test Action Group (JTAG), Universal Asynchronous Receiver/Transmitter (UART) and Serial Peripheral Interface (SPI)). Debugging interfaces might non-intentionally remain opened after manufacturing. We intend to observe how many measures were applied to these wireless nodes.
- Recover the cryptographic keys stored in the internal memories of the MCU, especially the JK which is used for the joining process in WirelessHART. Find out where the JK is stored and propose attacks to recover it.
- Measure how difficult the key retrieval could be for a real attacker. Provide different viewpoints from different attacker's profiles and find out how much time it can take to successfully complete a node capture attack.
- Perform an exhaustive analysis of the WirelessHART standard and focus on the security measures applied at different layers of the standard.

- Attempt to clarify the WirelessHART key management and especially the joining process where the JK is used.

1.3 Research Method

In this thesis, the node capture research has been carried out in a secure environment without interfering with any real activity in industrial plants. A WSN has been set up at our laboratory and several attack vectors have been run in exactly two kinds of wireless nodes. Although this thesis involves two brands, only all the details of one brand will be addressed. The other one will be addressed as `unknown-node` and the majority of its details will remain out of the public version. A Non Disclosure Agreement (NDA) has been signed between the different parties in order to conceal any information that could cause damage to these particular WSNs. This research does not intend to provide all the details but either is not certainly encouraging "Security through Obscurity". The author of this thesis is completely opposed towards this philosophy of security. Although this thesis does not reflect all the information analysed during my MSc thesis, the most relevant information is incorporated.

This research consists of a *black-box reverse engineering* without any tip from the manufacturer. The purpose is to figure out whether a motivated attacker with non-expensive equipment and well-known tools is capable of recovering the cryptographic keys. This thesis has been completed with the help of hundreds of data-sheets, papers and other similar research on WSNs. The workload of this research has consisted of approximately three and half months of security evaluation both practical and theoretical and also around two months of writing.

1.4 Outline

In this thesis, *Reverse Engineering on WirelessHART devices* research is addressed. It is intended to provide not only a theoretical approach, but also to address low-cost attacks on WSNs, such as node capture and the use of debugging tools.

The remainder of the chapters of this thesis are structured as follows. Chapter 2 presents a extensive background regarding WSNs and WirelessHART. Several protocols are dissected as well as the cryptography and key management in WirelessHART protocol. Chapter 3 presents a brief discussion about hardware security. Chapter 4 comments the hardware and software tools that were used during this thesis. Finally, Chapter 5 presents the targets addressed in this thesis and explains attacks applied to them. Chapter 6 has final conclusions and possible future work.

Chapter 2

Background

This chapter attempts to provide enough background for readers in order to properly understand the rest of chapters and mainly be able to follow the physical attacks carried out in this paper. First of all, an extensive introduction of WSNs and their applications is addressed. Then, is an explanation of what a wireless sensor node is comprised of and its components. After that, the HART and the IEEE 802.15.4 standard are dissected as well as the WirelessHART protocol. The latter is focused on its security, especially the cryptography and key management involved in this protocol. Finally, other similar wireless protocols based on the IEEE 802.15.4 standard are mentioned in comparison with WirelessHART.

2.1 Wireless Sensor Networks Applications

To properly understand the architecture of a WSN, we must be aware of the requirements in the real world. Nowadays, wireless sensor networks are widely used. Some of the most common uses of these networks can be seen in different scenarios. WSNs can be deployed in a number of different situations. Below, we address some usages and discuss why WSNs are so indispensable and necessary nowadays. It is important to indicate to readers that this section covers neither all the previous related works nor the most state-of-the-art techniques in WSNs. Nonetheless, many state-of-the-art techniques are named and discussed along with enormous applications of WSNs.

2.1.1 Smart-houses

People with reduced physical functions or elderly can be monitored thanks to intelligent sensors [3]. Modern sensor-embedded houses, also known as smart-houses, can provide assistance without limiting or disturbing the resident's daily routine, giving him/her well-being, comfort and pleasure. *Telemedicine* or *Telehealth* are two state-of-the-art terms which reflect solutions for many difficulties that disabled people experience. WSNs are successful at disease prevention, optimal control of home appliances (ex. heating, lighting, air conditioning and ventilation), monitoring chronically ill patients and motion detection to control patient activity. Smart-houses will undoubtedly become part of our future housing and help us endure diseases.

In 2007 [4], authors contributed to extending the WSNs usage for home health care and patient monitoring. First of all, they show some prototypes which could be integrated into existing infrastructures. Secondly, they find out opportunities for health care monitoring applications and eventually discussed why health care at home is so important and what the future would hold. Between the prototypes, we can find 'SleepSafe' which aims to avoid infant deaths by detecting the way infants are sleeping. There are studies which claim that infants sleeping on their backs reduce the occurrence of Sudden Infant Death Syndrome (SIDS) by at least 40%. A couple of sensors on the baby could improve detection of this problem. 'Baby Glove' is another prototype that is

aimed at issues with possible infant deaths involving low temperatures and hypo- or hyperthermia being the most common cases. 'Baby Glove' consists of a swaddling baby wrap with a couple of wireless sensors to monitor their temperature, hydration and pulse rate. 'FireLine' is a wireless heart rate sensor that can be used to decrease stress and injuries through real-time fire-fighter health monitoring. 'Heart@Home' is blood pressure monitor and tracking system to remedy heart diseases. According to doctors, tracking blood pressure daily is possibly the best way to fight against this disease. 'LISTSENse' is another prototype that empowers the hearing impaired with measurement of critical audible information in their environment.

2.1.2 Smart-cities

Although wireless sensors cannot be easily seen in our cities, they are spread out everywhere, either to oversee possible accidents, resource saving or monitoring to assure things function properly. It is important to address what objectives need to be covered to understand better how convenient WSNs are.

Structural Health Monitoring (SHM)

SHM is a sensor-based preventive approach which enables public safety by ensuring that the largest structures found in cities including buildings, bridges and roads are sound. A crack sensor could warn about an imminent catastrophic demolition and save thousands of lives.

Around 2003 [5], American and Japanese Researchers developed a wireless sensing unit for real-time structural response measurements using *MICA* [1] motes as a wireless sensor. During this time, the results showed to be promising but it was still a very new field to investigate.

In 2004 [6], WSNs only relied on data acquisition systems. *Wisden*, a WSN for SHM, emerged with new novel features such as a reliable data transport using a hybrid approach of end-to-end and hop-by-hop recovery and low-overhead data time stamping that did not require global clock synchronization.

In 2007 [7] a WSN was deployed and tested on the 4200ft long main span and the south tower of the Golden Gate bridge in the American district of California. In this study 64 nodes were distributed over the main span and the tower, resulting in reliable sampled data even with 46 hops in the WSN. This research covers an accurate data acquisition system and high-frequency sampling with time synchronized sampling that was not apparently provided for previous research at the time of this survey. Authors claim that small packet size was a bottleneck for network data transmission, but increasing packet size can only be carried out when the wireless sensors owns a decent amount of RAM. During this study, *MICA* motes were used. This deployment seems to have been the largest WSN for SHM in those days.

Around 2008 [8], an accelerometer sensor node was designed, calibrated and developed to satisfy the requirements for structural vibration monitoring and model identification. The SHM is based on TinyOS [2] operating system to provide a flexible software platform and scalable performance. Regarding the hardware side, a microcontroller for processing each node has around four bidirectional channels and Radio Frequency (RF) device for communicating wirelessly. This prototype was deployed on a long-span bridge with 64 nodes obtaining accurate ambient vibration data for identifying vibration modes of a bridge.

In 2011 [9] the Jindo Bridge, a cable-stayed bridge located in South Korea, was fitted with sensors, improving the previous ones from one year before. This implementation encompassed the following unique features: the world's largest WSN for SHM to this date, power harvesting

[1] http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf
[2] http://tinyos.net

enabled for all sensor nodes, a better and more reliable data acquisition scheme and decentralized data aggregation; making the WSN much more scalable to a large and dense sensor network, environmental monitoring and decentralized cable tension monitoring.

In 2012 [10], author carried out some state-of-the-art techniques in commercial aviation in order to improve economics and safety. The number of ageing aircraft in service is increasing and many intensive inspections must be performed to ensure safety. An ideal SHM would be able to provide accurate details about damage type, location, severity and also estimate the remaining life of the structure while the structure is still in use.

In 2012 [11], a wireless sensor system was tested with many of the commonly used types of sensors in suspension bridges to prove the viability of wireless sensor network in actual implementation. Whereas *ZigBee* is used for short-distance communications among sensors, Code Division Multiple Access (CDMA) for long-distance wireless communications is used with remote locations.

Garbage level

This feature could enable waste management services to empty the rubbish only from bins that are sufficiently full, saving time and gas. Furthermore, with this promotion of public health by enabling timely garbage collection, we can avoid bad odours and garbage overflows.

In [12], the authors propose a WSN architecture for solid waste management. Using some known wireless sensors, *TelosB* [3], and data transfer nodes, authors offer the possibility to observe the garbage bins' measurements monitored by accessing with a normal web browser.

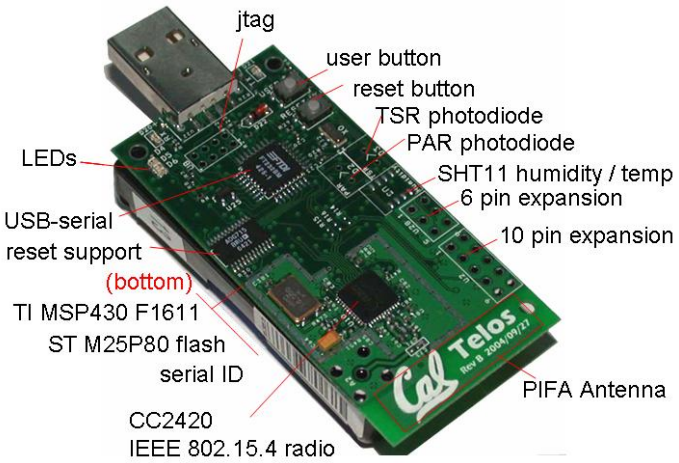


Figure 2.1: Physical specifications of a Telos Revision B. 2004

On the other hand, in 2014 [13], authors combined WSNs with Vehicular Ad-Hoc Networks (VANETs). Principally what this system addresses is that garbage collecting vehicles are being dynamically informed to clear the dustbins in real time. There is a base station that monitors the entire garbage collection and disposal module. Moreover, databases are constantly updated and it is possible at any time to know the status of dustbins and the location of garbage collecting vehicles.

[3] http://www4.ncsu.edu/~kkolla/CSC714/datasheet.pdf

Traffic and Parking management

By providing accurate information on available parking spaces, motorists save time and fuel and cities reduce atmospheric pollution and congestion. Besides, though traffic control systems, we can relieve traffic congestion in big cities or very crowded spaces. But not only traffic jams can be minimised, moreover it also helps to improve the reduction of CO_2 emissions.

Since expanding parking area is extremely expensive and unfeasible, between 2007 and 2013 [14]-[15], researchers proposed a smart parking system combining Radio Frequency Identification (RFID) and Zigbee technologies. The system can inform drivers about the amount of available parking areas and in which areas they should be redirected to.

Various previous works can be found in 2012 [16], where authors dissected various state-of-the-art smart parking systems and offered comparisons between them. They focused on which sort of sensor was used, whether they are using a central server, possibility for detection of different objects and parking methods. In this paper, a costumer would be able to determine space availability and the parking operator could predict future parking patterns and trends. This would help avoid vehicle thefts and eventually improve parking timing for customers.

In 2013 [17], authors showed some applications and architectures used. Some of the aims represent for instance the parking management, traffic light control, traffic optimization, safety or pollution prevention. Authors address a study of these goals and compare all of them with previous related work.

Noise pollution

Noise pollution is a common environmental problem affecting cities that can be detected by using noise sensors. These are suitable for creating a real-time noise map of cities and improve the life and health of people.

In 2007 [18] a survey is shown using traditional WSNs and obtaining decent results. The authors use well-known hardware, such as *Tmote Sky* [4], as development hardware platform for their research. Measuring noise might be a serious problem due to many human beings have very different perceptions. Microphones convert pressure fluctuations into an equivalent electrical signal which can be processed to compute the loudness of the noise source that generated the acoustic wave. Average loudness levels over long periods of time are commonly used as noise indicators.

In 2010 [19] another interesting proposal and apparently cheaper way of noise pollution monitoring can be carried out with smartphones. Nowadays, almost everyone has a smartphone near him/her most of the time and these devices have all the necessary functions to act as wireless sensors. Access to sensors, accelerometers, Global Positioning System (GPS), wireless devices and connection to the Internet is all available to us right from our pockets. Kanjo proposes *NoiseSpy*, a sound sensing system that turns the mobile phone into a low-cost data logger for monitoring environmental noise.

In 2013 [20], Australian researchers proposed a pilot study of urban noise monitoring architecture using WSNs and a new noise monitoring hardware platform which processes the analogue noise signal and converts it into sound levels ensuring privacy protection. In order to visualize the data, Google Maps is used. This architecture attempts to be scalable and applicable to other sensors required for city management.

[4] http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf

Atmospheric pollution

Wireless sensors can measure the level of dust, CO_2, NO_2 and SO_2 to avoid respiratory diseases. Big cities are already using more and more of these sensors to face the pollution and try to keep it below damaging levels.

In 2010 [21], a Wireless Sensor Network Air Pollution Monitoring System (WAPMS) was built in the Mauritius island. This innovative system claimed to notably reduce power consumption by proposing a new recursive data aggregation algorithm. This algorithm merges data and filters out invalid readings, sending only the right data to the main re-collector, resulting in energy saving.

In 2011 [22], a real-time WAPMS was offered to the community. In this research, authors designed a WAPMS which was deployed in the city of Hyderabad. They used multi hop data aggregation algorithm to deal with data gathering and routing problems. Mainly, the pollution levels of CO_2, NO_2 and SO_2 were analysed on this survey. In this paper a lightweight middle-ware is developed along with a web interface to observe the live pollution data in the form of charts and numbers.

Recently, in 2014 [23], another WAPMS was proposed through a Mobile Data Acquisition Unit and a fixed Internet-enabled Pollution Monitoring server to recollect data. Basically, this WAPMS relies on a mobile re-collector which gathers all the pollutant levels of CO_2, NO_2 and SO_2 and then packs this data along with its GPS location into frames. These frames are transmitted towards the central-server via the Zigbee module. The server is interfaced to Google Maps to display the location of the hardware unit.

Another research in 2014 [24] reviewed the importance of employing WSNs in air pollution monitoring in cement factories. In a cement factory, air pollution monitoring systems very rarely use wireless communications. There exist various well-known wired systems such as *Opsis* [5] and *Uras26*. However, these systems just monitor the emissions from the chimney, leaving numerous areas without coverage. Nowadays, many surveys are being constantly carried out and analysed to progress in this field.

2.1.3 Habitat monitoring

Habitat monitoring is an important tool for assessing the threat and conservation status of species and protected areas. The focus on the protection and preservation of the environment has been steadily gaining relevance in scientific communities and society. WSNs represent a big impact for this purpose due to how helpful it is for the efforts to reduce disturbance effects, predation of endangered species and the shift to unsuitable areas.

In 2002 [25] a specific habitat monitoring application for a small island using 32 nodes was developed. This system architecture could serve numerous uses for habitat monitoring. This survey showed the hardware requirements: sensor platform, enclosure design and sensor calibration, along with constraints and guidelines that served as a basis for a general wireless sensor architecture for many such applications. Ultimately, in the paper they monitored a sea-bird nesting environment and its behaviour as an example of their application.

In 2010 [26], a similar case was carried out for monitoring sea-birds on the Skomer Island, a United Kingdom national nature reserve. One more time, this paper contributed to the community offering insights on design decisions, problems found and experience gained.

[5] http://opsis.se

2.1.4 River monitoring

In 2010 [27], Glatz et al. proposed the first monitoring wireless network using ultra-capacitors as energy storage elements to compensate for the irregularity of environmental energy sources. Instead of using self-powered sensors they relied on energy harvesting to do a perpetual operation. Glatz et al. dissected a variety of low-power micro-controllers to find the best results focussed on energy harvesting.

In 2013 [28], more surveys have been carried out in Brazil due to climate changes. This study showed the city of São Carlos during the rainy season floods to protect the critical places which were vulnerable to the risk of sudden downpours. In this paper, authors attempted to stem the possible flooding and monitor the river water level. Authors claimed that the problem of only an information source, the geo-spatial information, could be notably improved with the use of WSNs. They made a real time river monitoring system, using web technology with various wireless sensors. The next step was to increase the number of wireless sensors to detect possible overflows. Furthermore, there are various projects on the Internet for building and deploying a real-time sensing infrastructure for environmental monitoring. An example of one is RiverNet [6].

2.1.5 Smart-Metering

The precise measurement of certain levels is becoming a key process in smart-metering. An example can be seen by measuring the water flow, electric current, weight of materials and goods, liquid levels, distance of ultrasounds and distance and displacement of an object. A handful of applications use smart-metering. Examples are electric and water consumption, pipe leakage detection, agricultural irrigation, tanks level control, industrial automation, etc. More industries are beginning to benefit from the use of wireless sensors networks.

In 2011 [29], Korean researchers proposed a test bed in Smart-grid systems, describing their implementations and outcomes in a field demonstration that monitors the usage and generation of electricity in a small building. Researchers deployed various smart meters, wind power generators, a photo-voltaic power generator, a battery, two electric vehicle chargers, two light controllers and a smart outlet. The light controllers exchanged their data and control messages through Programmable Logic Controller (PLC) and the other devices. The communication was carried out through wireless sensors, mainly the ZigBee technology.

In 2011 [30], developments on WSNs for detection of combustible or explosive gases was an emerging topic. Russian researchers learned to address the early gas detection with the help of a board 2D semiconductor sensor by generating a Wireless Gas Sensor Network.

In 2013 [31], Italian researchers described a new prototype low-cost WSN for gas smart-metering. The authors relied on a star topology network where each node could be integrated with a standard relay gas meter. These nodes could measure the gas usage and send it to the central node in the star. All this information was carried out through RF links. Whereas the wireless sensor nodes were self-powered, the central node was connected directly to the electric power. The information was processed in a Personal Computer (PC) connected to the central node. A framework was responsible of intercepting and processing all data allowing authorized users to check the network status through a web interface. More wireless nodes could join the wireless network due to the self-learning of the WSN.

In 2013 [32], a study on fluid leakage in pipelines using WSNs eventually addressed issues with four different solutions using WSNs. The WSN-based solutions were magnetic induction based, continuous pressure monitoring, underground to above ground radio propagation and wireless signal networks. Ultimately, they concluded that after introducing the theoretical structure for

[6] http://sensors.cs.umass.edu/projects/rivernet

magnetic induction-based, deploying this system in real-life needed much work. Advantages and disadvantages were always present, however these techniques were really useful to avoid pipelines leakages due to their real-time detection.

In 2014 [33], authors proposed a transient pressure wave based technique coupled with wavelet analysis to achieve reliable detection and localization of abrupt bursts and leakages. They concluded the paper presenting an effective algorithm, based on the information carried in the transient pressure signal, which obtained more than 90% accuracy for hazardous situations. Similar research has been carried out in [34] (2006), [35] (2007), [36, 37] (2008) and [38] (2011).

2.1.6 Warfare: Tracking enemies and tactical techniques

The numerous military and civilian applications of this technology have the potential to make a considerable impact on society. The main goal of surveillance missions is to obtain and verify information about enemy capabilities and positions of hostile targets. Such missions often involve a high element of risk for human personnel and require a high degree of stealthiness. Therefore, the ability to deploy unmanned surveillance missions by using WSNs is of great practical importance for the military industry. Another possible usage of WSNs has been carried out by The DARPA project [7]. Such projects have used WSNs for detecting minefields in the battlefields with satisfactory results.

In 2004 [39], Americans carried out various experiments using ad-hoc WSNs to perceive enemies. This system was capable of detecting and accurately locating shooters even in urban environments. A group of wireless nodes were spread out and they communicate as an ad-hoc WSN. The system was reliable even with multiple sensor failures while also offering proper coverage and high accuracy. This paper covered acoustic signal detection, the most important middle-ware services and the unique sensor fusion algorithm. The system performance was analysed using real measurement data obtained at a US Army MOUT (Military Operations in Urban Terrain) facility.

In 2004 [40], American authors described and designed an implementation of a running system for energy-efficient surveillance. The experiment counted around seventy *MICA2* [8] motes forming a WSN. The system allowed a group of cooperating sensors to track and detect the locations of moving vehicles in an energy-efficient and stealthy manner. Unlike other works, this paper claimed to have been tested in outside settings. Furthermore, authors implemented an entire integrated suite of protocols and application modules. Ultimately, authors concluded by explaining the key lessons learned; both hardware and software design must not be ignored in developing usable solutions. This includes realism of sensor performance, asymmetries in communication, false alarms, and race conditions.

In 2004 [41], several American universities studied the intrusion detection in what they called 'A Line in the Sand' security scenario. Researchers explored the design space of sensors, signal processing algorithms, communications, networking, and middle-ware services. A contribution of their work is that they did not suppose a reliable network; on the contrary, they totally assumed the unreliability of the WSN. This experiment was deployed and tested at MacDill Air Force Base in Florida. Furthermore, less than one hundreds motes were used but the authors articulate a few of the challenges facing extreme scaling to tens or hundreds of thousands of motes for a possible occasion in the future.

In 2006 [42], a survey investigated the use of WSN technology for ground surveillance by Australian researchers. The focus of this approach consisted of being able to deploy the system outdoor, detect multiple targets such as tanks or troop movements and use inexpensive off-the-shelf

[7] www.darpa.mil
[8] http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf

wireless sensor devices. Such devices must be able to measure sensing acoustics and magnetic signals generated by the enemies' devices. This paper related how extremely challenging real-time tacking is and how real-time decision making needs to be addressed. They proposed and emphasized a Hybrid Sensor Network Architecture (HSNA) tailored specifically to address such requirements.

In 2009 [43], American researchers from Boston outline different application scenarios in remote large-scale WSNs focusing on the primary requirements for tactical environments. Authors propose a WSN architecture based on the cluster-tree based multi-hop model with optimized cluster head election and the the corresponding node design to meet the tactical requirements.

2.1.7 Natural disasters

Wildfires have been extinguished with the help of wireless sensors dropped from aircrafts overhead. These sensors help determine temperature measurements and create a temperature map. Furthermore, many scientific studies use wireless sensors to predict volcanic eruptions. Such examples are not comparable with the number of applications we can see today working around the globe.

In 2006 [44] a forest fires surveillance system was addressed by Korean researchers. Basically this system utilizes a WSN, middleware and web application to detect possible forest fires. Previous investigations were using surveillance cameras with a combination of infra-red sensors system and satellite system. As the authors claim, these systems can endure no real-time applications.

In 2006 [45] scientific researchers lead a study to collect seismic and infrasonic (low-frequency) signals in volcanoes. Previously in 2004, this group had installed some tiny wireless sensors in the Volcán Tungurahua in Ecuador as a proof of concept. Later in 2005, [46]-[47] in another Volcán Reventador also located in Ecuador, the scientific researchers deployed 16 sensor nodes equipped with a microphone and siesmometer collecting seismic and acoustic data on volcanic activity. As many of the WSNs, this WSN comprised of wireless sensors relaying data via a multihop network until reaching a gateway node. This gateway node was connected to the Internet through a modem remotely allowing connectivity from a laptop. A GPS receiver was used along with a multihop time-synchronization protocol to establish a network-wide timebase. This paper showed the design reflections when they chose the network hardware. Ultimately, they decided to use well-known *TMote Sky* [9] instead of *TI MSP430* MCU for storage capacity. The outcome was greatly successful even having some outages during some days. Scientific researchers were able to collect a great deal of information and capture volcanic events based on the acoustics.

2.1.8 Agriculture and animals tracking

WSNs have arrived to all corners of industry and agriculture and animal tracking is not an exception. Agriculture faces many challenging problems such as climate change, water shortages, labour shortages due to an ageing urbanized population and countless other reasons. Humanity depends on agriculture and water and other factors for survival. Regarding animal tracking, it might be applied for pastoral tracking or for controlling wildlife. In the *ZebraNet project* [48], wireless sensors combined with GPS technology has been used to observe the zebra's migrations. These experiments with animals have also been seen with falcons and salmon without the use of GPS technology.

In 2004 [49], Australian researchers described some new wireless sensor hardware developed for pastoral and environmental applications. The paper explains the necessities of developing their own hardware due to the problems with the radio range of the current hardware. One of the

[9] http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf

requirements of this industry was the support for solar cells, a proper radio range and a right and resistant connector between the board and the RF module. At the time of this research, there was not as much hardware available as today. Therefore, neither *MICA1* [10] nor any other wireless sensors of the time fulfilled the requirements. They released a new family of wireless sensors with the required improvements and called them: 'Flecks'. These motes were inspired by Berkeley. During research, different versions of 'Flecks' devices arose. Principally, the devices comprised an *Atmel Atmega128* processor and a *Nordic* [11] radio chip. Newer versions were adding GPS and more storage in MultiMedia Card (MMC) flash memory generating 'Flecks1' and 'Flecks2' versions. These experiments allowed researchers to investigate in pastoral tracking.

In 2007 [50], researchers one again came up with a newer version of 'Flecks'. Whereas 'Flecks1' and 'Flecks2' were operating on the $433Mhz$, the newest 'Flecks3' was operating on the $915Mhz$. 'Flecks3' also includes an internal clock to avoid overheads and integral solar battery due to the sunny environment in Australia. Knowing the state of pastures and crop fields in a farm environment is crucial for farmers. From crops mature to cattle graze pastures for food, farmers must decide when to irrigate pastures, apply fertilizers or move cattle to different pasture. Normally, farmers rely on a combination of experience, visual observation and intuition to make such decisions, however they will be nowhere near to the optimal one. This research attempted to model herd and individual behaviour by wearing 'Flecks3' collars. After this research, it was feasible to record where every mouthful of grass has been taken from, as well as where and how quickly future pasture growth could occur.

2.2 Sensor node Architecture

This section tries to mainly analyse the parts a sensor node is comprised of. Although every sensor node can be tailored according the requirements, eventually all sensor nodes have similar components. This section is going deeper into each part of wireless sensor nodes as well as the most known wireless sensor nodes. A little journey in wireless sensor nodes is addressed in this section. A wireless sensor node is basically comprised of a main and low-power MCU to process all operations, a RF module in order to carry out wireless communications, an optional external memory to log data in it and batteries or any source of power supply to stay alive WSNs for a long lifespan.

In Figure 2.2 is depicted an internal representation of a common wireless sensor node. Although there exist different combinations of designing wireless sensor nodes. Figure 2.2 reflects the most popular representation.

2.2.1 Microcontroller (MCU)

The Internet Of Things (IoT) world has arrived with tens of trillions of embedded devices around the world. Such devices are comprised of different MCU's families and each one of them carries out a number of applications. Nowadays, we carry a smart-phone with us everywhere. Those smart-phones are formed by many MCUs which are able to measure temperature, record audio, take pictures or videos, measure pressure or acceleration and countless other things. We are constantly surrounded by tiny Central Processing Unit (CPU)s. The IoT is foreseeing an enormous amount of new devices in the upcoming years. Unlike IoT, the WSNs are not all connected over the Internet. Although WSNs can operate through Internet Protocol version 4 (IPv4) or Internet Protocol version 6 (IPv6) indistinctly, for security reasons only the gateways are normally connected to the Internet.

A MCU, also known as μC or uC, is a tiny computer on a single Integrated Circuit (IC). Such a chip has an internal processor core, an on-chip Flash and Random Access Memory (RAM) mem-

[10] http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf
[11] http://www.nordicsemi.com/eng/Products/2.4GHz-RF

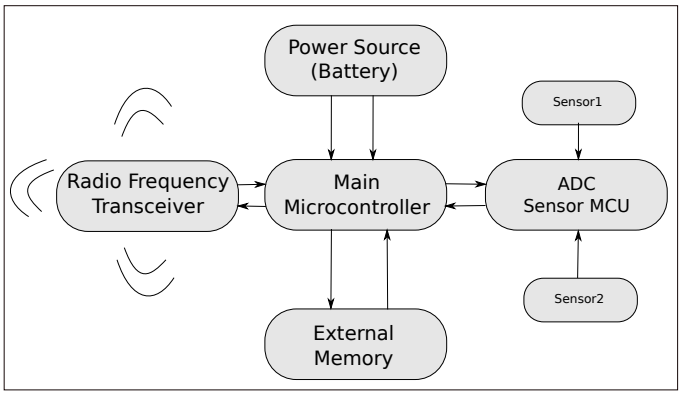


Figure 2.2: General architecture for wireless sensor nodes

ory as well as other features depending on its manufacturer and scope. MCUs are mainly designed for embedded applications. In the MCU's world we can observe a classification of architectures based on the number of bits. MCUs are classified by families and then organised according to the #bits. Like the CPU, the MCU operates with words formed by a certain fixed amount of bits depending on their architecture. MCUs also run at certain clock frequencies relying on their hardware specifications to implement a set of machine instructions.

When WSNs manufacturers select their embedded MCUs, they face many problems to choose the most adequate design to fulfil the WSNs requirements. Power consumption is perhaps the most critical feature in WSNs that designers take into account before designing. The less power consumed, the longer the lifespan is for the wireless sensor node. The physical security needs to be discussed in detail and the MCUs must have at least sufficient anti-tampering protections to come out unharmed after the most common physical attacks. Nonetheless, the performance is very important as well. More important characteristics are speed, area, storage size, computing power, maturity of the debugging tools, Application Programming Interface (API) available and so on.

As has been explained before, a wireless sensor node has a main MCU which is responsible for processing all data and sending it to the RF module to be transmitted. On the other side, when the data is coming from the RF module from other wireless sensor nodes, the data needs to either be just forwarded or preprocessed again and sent back to the next hop. Although sometimes, there is not only a MCU, but there is always a MCU that acts as the principal one.

It is possible to combine all the requirements of a wireless sensor node in only one IC. For example, we can combine a main MCU, RF technology, RAM memory, a sensor MCU and on-board flash memory. Although this approach is not the most commonly used worldwide, it might be encountered in the industry of WSNs.

2.2.2 Radio

An IEEE 802.15.4 compliant device can be attached to a module, just the transceiver itself or a single-chip with both the RF and main MCU. In this section some RF modules are discussed but the same information is attained to transceivers. The RF device is responsible for wirelessly transmitting all the information in a WirelessHART communication. All the information is sent from the main MCU to the RF chip through commands. Such commands are dependent on the

manufacturer of the RF module. When a manufacturer sells these RF modules, an API is mandatory in order for it to be easy and straightforward for re-seller's developers. An API is a set of instructions or commands with which a RF MCU talks to another MCU or other component. This API is normally accessed through a UART interface with the main ports: `TX` (Transmission) and `RX` (Reception). Depending on the manufacturer, different extra ports might be important to establish a communication with the RF. The UART speed in a communication is also dependent on the manufacturer; although it is normally either 9600 bits per second (bps) or 115200 bps. Once the communication is available and the right pins are well connected, a command-response can be started.

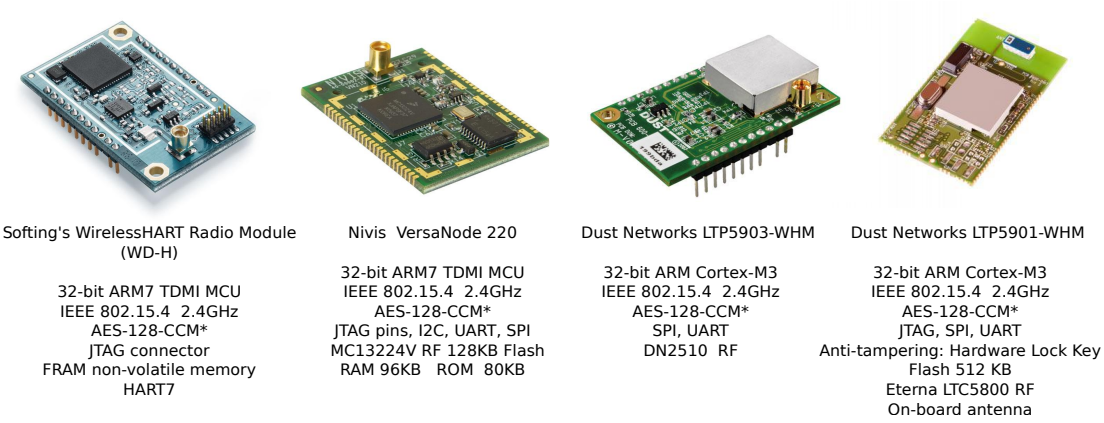


Figure 2.3: Comparison between WirelessHART Radio Modules and their main specifications.

The RF module in a wireless node can appear in different ways: integrated on the main MCU, in an external module, attached to the main board as another component or just simply as a mote itself with all of it integrated in one MCU. Figure 2.3 depicts some of the WirelessHART RF modules and their key characteristics. Figure 2.3 mentions the main architecture of the WirelessHART RF modules on the market, their programming and debugging interfaces, their anti-tampering protections, whether they are mentioned on the datasheet and their non-volatile memory to save persistent data. All the RF modules own an on-board Advanced Encryption Standard (AES)-128 hardware accelerator responsible for encrypting and decrypting all the communication with the other sensors nodes.

As is shown in Figure 2.3, there are specific manufacturers of these MCUs. Some well-known chip manufactureres are Atmel, Atmel AVR and Texas Instruments (TI). For Atmel RF modules we can highlight:

- *Transceivers.* Some reference: AT86RF231, AT86RF232, AT86RF233 at 2.4GHz and AT86RF233 at 700/800/900MHz.
- *RF module.* Being a combination of main MCU and RF chip in an extra module. Some references : AT86RF212B ZigBit, AT86RF233, ATmega256RFR2, ATxmega256A3U and AT86RF212B Module, ATxmega256A3U and AT86RF233 Module.
- *Single-chip.* Being main and RF MCU in the same chip. The architecture is based on ARM Cortex M0 (RF) and AVR 8-bit as main one. References: ATSAMR21E16A, ATSAMR21E17A, ATSAMR21E18A, ATSAMR21G16A, ATSAMR21G17A, ATSAMR21G18A.

Likewise, TI has a lot of different chips depending on the wireless protocol, focusing on IEEE 802.15.4 and WirelessHART and *ZigBee*, TI offers the following principal references: CC252x, CC253x, CC263x and MSP430.

Sensors

Each mote or wireless node has one or several sensors attached to it. There exist different kinds of sensors depending on the application. Mainly, the most used sensors are temperature, pressure, movement, density, distances, noise and vibration, amongst others. At least one sensor is always on-board in wireless nodes. Being possible to stay integrated either in the same RF module, in the main MCU or being a dedicated MCU for acquiring measurements from the environment. When a sensor is included in the same RF module, it is normally called a Wireless MCU. It can act as a wireless sensor node by itself. Since some Atmel and TI MCU were already described in the RF modules section (2.2.2) and sometimes sensor MCUs are integrated into others chips, it is not considered relevant to dive into.

2.2.3 External Memory

Many wireless sensors are constrained by their resources and are therefore not capable of storing all their measurements in their on-chip memories. Although MCUs have an internal flash or non-volatile memory, it is not considered a good idea to store data in these memories. First of all because wireless sensors have a reduced lifespan depending on their write operations. Secondly, because as the monitoring is saving data continually, the majority of wireless sensor nodes are equipped with an external and persistent memory. In this memory, a bunch of measurements are stored and they remain available after switching off the wireless node. Normally these external memories are serial memories or serial EEPROMs. Such memory devices communicate with the MCUs through the SPI or Inter-Integrated Circuit (I2C) interfaces. Both SPI and I2C are two serial hardware protocols based on Master-Slave communication and are consulted via command-response protocol.

2.2.4 Power source

There are at least three known current sources: autonomous batteries, energy harvesting with solar panels and the most unusual ones connected to the plug. Wireless sensor nodes are well-known for being self-powered and freely spread out into the world. This fact falls on the usage of batteries as manner of keeping the nodes alive. Since the low-power MCUs are specially designed to cut down the consumption, MCUs are designed to stay asleep as much as they can when there is no RF communication or serial commands from the main MCU. The node's lifespan is estimated to be up to 10 years, although there exist nodes that just live for around 5 years. When conditions are harsh, the lifetime is drastically reduced to approximately 2 years. The most popular batteries used in the wireless sensor nodes are either button batteries or a couple of the well-known AA Mignon batteries.

2.3 HART

Before talking about some wireless protocols, it is convenient to introduce protocols which are encapsulated through wireless technology. Once we have reviewed the details of such protocols, the wireless technology will be easier to comprehend.

2.3.1 HART Protocol

HART [51] is an industrial and low-level network protocol which operates through $4-20mA$ current loop both analogically and digitally for mainly smart field devices. Nowadays, the HART technology is the most widely used field devices communication protocol for intelligent process instrumentation. In the late 1980s, a company called 'Rosemount Inc.' developed HART as proprietary protocol for their field devices, principally based on *Bell-202* protocol [52]. Around 1986, it was released as an open protocol which has become mature through constant improvements. However, the revolutionary idea was to combine HART with the wireless technology with the

purpose of reducing costs and reaching places which were extremely expensive or thought to be impossible. This resulted in the birth of the WirelessHART protocol. The HART protocol is used mainly in power plants, chemical factories and oil and gas industries. This protocol has neither authentication, authorization nor encryption support.

2.3.2 Modulation: Audio Frequency Shift Keying (AFSK)

As we mentioned before, HART operates over the $4-20mA$ current loop. Furthermore, the HART protocol is based on the *Bell-202* [52] modem communication standard. The *Bell-202* modem data modulation uses the AFSK to encode data at a rate of 1200 bps. Basically, *Bell-202* transfers serial binary data by using frequency manipulation. The one and zero are encoded by using different harmonics of the frequency range. *Bell-202* AFSK uses a 1200Hz tone for mark, also interpreted as unit or a binary '1', and 2200Hz for space interpreted as '0' in binary format. According to Figure 2.4, the analogue signal is propagated with a different wave amplitude, resulting in a digital signal constituted of ones and zeroes. One important thing to mention is that analogue and digital in the same channel, allows to monitor the Process Variable (PV) from slaves to masters without any interruption. Moreover, various digital readings can be carried out per second. As the digital AFSK signal is phase continuous, there is no interference with the analogue $4-20mA$ signal and two simultaneous communication channels are feasible.

Both handheld devices and field devices have an internal AFSK modem which is used to interact on the analogue channel, whereas the PC stations must use a serial interface to connect the AFSK modem.

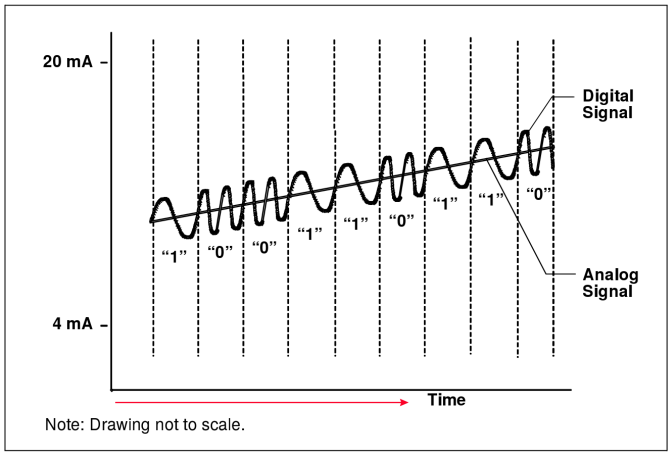


Figure 2.4: AFSK. Simultaneous Analogue and Digital Communication

2.3.3 Communication Modes

In the HART protocol we can distinguish a couple of communication modes:

- Master-Slave mode. This mode is a basic master-slave communication protocol where master devices, such as Distributed Control System (DCS), PLC, PC or a handheld device, start the communication with slaves (smart field devices). Another option can be addressed as master-master communication, where there is a primary master, normally a DCS, PLC or PC, and a secondary master being a PC or a handheld device. An example can be viewed in Figures 2.5 and 2.6.
- Burst mode. Some HART devices can additionally support this mode. The Burst mode speeds up the communication. The master instructs and enforces the slave in order to

constantly broadcast a HART response. Therefore, the master is capable of receiving the HART message at a higher rate until it instructs the slave to stop bursting.

Generally speaking, a HART loop is a communication network in which the master and slave devices are HART compatible. A HART loop can be comprised of slaves and masters. In order to start a HART communication, it requires at least one master device to start the communication. When the communication has started, masters are able to send HART commands to the slaves and receive responses.

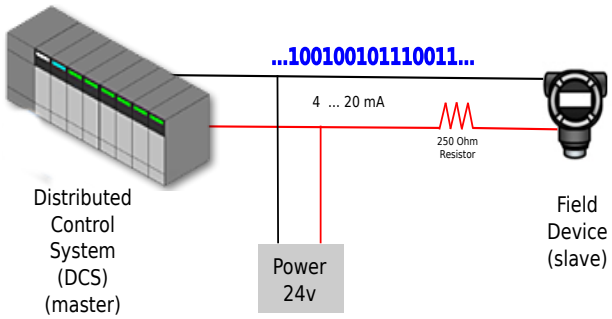


Figure 2.5: HART Master-Slave communication between a DCS and field device.

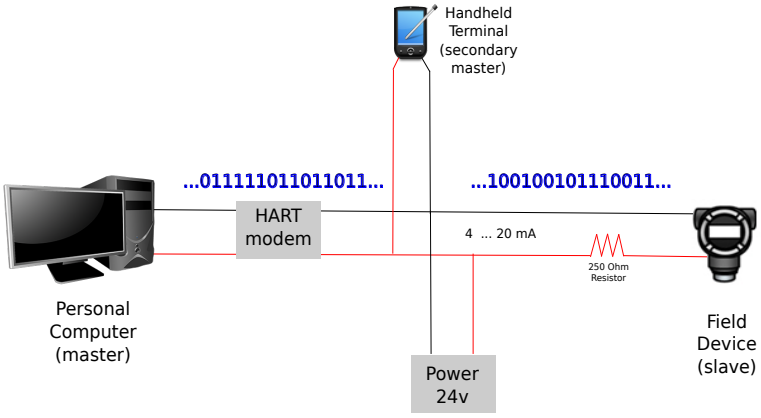


Figure 2.6: HART two masters communication between a PC, handheld terminal and field device.

2.3.4 Network Configurations

Principally, HART communications are addressed according to addresses of each device. Such device address allows us to drive traffic and be able to route data through it. In the HART protocol we can distinguish two sorts of network configurations:

- Point-to-Point (Analogue and digital). This modality involves two devices such as a field device(slave) and a PC or handheld terminal(master). By using the HART protocol, both communication channels transmit different data. Only a PV is used in the analogue $4 - 20mA$ signal. The digital signal is communicating with the rest of data in the form of PV, configuration parameters or other device information. A condition must be respected; the device address of the field device needs to be zero in order to be able to establish communication.
- Multidrop (Digital). Only the digital signal is used while the analogue is fixed to $4mA$ to provide a current loop. In this mode, up to 15 devices can be connected in parallel to a single wire pair. The PC host, master in the communication, is capable of assigning different device addresses within the range of 1 to 15. Newer HART revisions allow addresses up to 63 and having each device a unique address. Unlike point-to-point, all the devices addresses are > 0.

2.3.5 HART communication layers

In Table 2.1 below, we see a comparison between the 7 layers of the OSI model and the HART communication layers. We address a brief detail about each HART layer to be able to understand how HART operates.

OSI layers	**HART layers**
Application	HART commands
Presentation Session	
Transport	End-to-end communications
Network	Routing
Data link	HART protocol rules. Master-slave protocol
Physical layer	Bell 202, AFSK-bus

Table 2.1: HART protocol and OSI model.

2.3.6 Packet Structure

The structure of a HART packet is explained below and it can have different variants depending on whether it is a master or slave packet, the command and its payload and so on. First we show the graphical view of a HART packet structure, depicted in Table 2.2, and after that we give a brief explanation of each field in the structure. Eventually, we provide a big picture with the combination of the field name, its length and a small detail in Table 2.3.

Preamble	StartByte	Address	Command	ByteCount	Status	Payload	Checksum

Table 2.2: Structure of a HART packet.

- *Preamble.* Between 5 and 20 bytes with the value of '`0xFF`'. It synchronises the stream before starting between participants.
- *Start Byte.* It may have several different values depending on the type of message: master to slave, slave to master or even burst message from the slave. It also contains the address format; short or long frame depending on the HART revision.

- *Address.* For a short frame format: It contains one byte with one bit serving to distinguish between two masters and another bit to point to burst mode packets. For a long frame format: It contains 5 bytes, therefore the field device ID is represented with 38 bits.
- *Command.* This field is represented by a 1 byte numerical value which encodes the master commands of 3 categories: Universal, Common-Practice and Device-Specific commands.
- *Byte Count.* This fields indicates the message length. This is the manner for the receiver to distinguish between the payload and checksum. The amount of bytes depends on the sum of the status and the payload bytes.
- *Status.* A couple of bytes are only included in responses from slaves and contain health status. Such bytes indicate whether the communication is successful or not. In the affirmative case, these bytes are zeroed on the slaves. Masters do not use these 2 status bytes.
- *Payload.* It contains data depending on the command to be executed. It could be empty depending on the command.
- *Checksum.* Also called parity, it is a eXclusive OR (XOR) operation containing all the bytes from Start Byte until the last byte of the payload and resulting into a final byte.

Summarizing, in Table 2.3 we can observe a big picture of how a HART packet seems.

Field Name	**Length (Bytes)**	**Purpose**
Preamble	5-20	Synchronization and Carrier Detect
Start Byte	1	Specifies Master Number
Address	1-5	Specifies slave, Specifies Master and Indicates Burst Mode
Command	1	Numerical Value for the command to be executed
Byte Count	1	Indicates the size of the Data Field
Status	Master (0) Slave (2)	Execution and Health Reply
Data (Payload)	0-253	Data associated with the command
Checksum	1	XOR of all bytes from Start Byte to Last byte of Data

Table 2.3: Dissection of a HART packet.

2.3.7 HART commands

The HART command set includes three classes of commands: universal, common practice and device specific. The *Universal* ones, as their name implies, are supported for all the HART devices. The Universal commands allows access to basic information such as primary variables and units. The *common practice* ones provide functions common to many field devices, but not all of them. Finally, the *device specific* ones are functions only developed in specific field devices and usually specified by the device manufacturer. Below, we offer some examples of the available functions on the HART commands according to their class. Further information and technical details can be found in the HART Field Communication Protocol Application Guide [51].

2.3.8 HART-IP

The 2-wires HART protocol has been extended to wireless mesh networks (WirelessHART) as well as Internet Protocol (IP) networks. The latter encapsulates the HART protocol packet structure into the IP protocol enabling the use of TCP and UDP protocols. The HART-IP extension

Cmd	Description
00	Read Unique Identifier. Uses the polling address to establish a connection with the field device
01	Read PV
02	Read Loop Current And Percent Of Range
03	Read Dynamic Variables And Loop Current. Reads dynamic variables, PV and current loop
06	Write Polling Address
07	Read Loop Configuration. Reads the polling address and whether the loop current is active or not
08	Read Dynamic Variable Configuration. Reads the type of each process variable
09	Read Device Variables with Status. Reads up to 4 process variables with data quality status
11	Read Unique Identifier Associated with Tag. Uses the 8-character tag to establish a connection with the field device
12	Read Message
13	Read Tag, Descriptor, Date
14	Read Primary Variable Transducer Information
15	Read Device Information. Reads upper and lower range values and other device related
16	Read Final Assembly Number
17	Write Message
18	Write Tag, Descriptor, Date
19	Write Final Assembly Number
20	read Long Tag
21	Read Unique Identifier Associated with Long Tag
22	Write Long Tag

Table 2.4: Examples of HART Universal commands

connects to the plant networking infrastructure, provides fast access to measurement and device diagnostics and enables enterprise-wide access to information. Any Ethernet network can be remotely accessed from anywhere in the world as a mixed network through wired and wireless devices. To summarise, the main scope of this protocol falls on the fast and reliable union of the enterprise-level with the WirelessHART gateways.

Cmd	Description
34	Write PV Damping Value
35	Write Primary Variable Range Values
38	Reset Configuration Changed Flag
40	Enter/Exit Fixed Current Mode
41	Perform Self Test
42	Perform Device Reset
44	Write Primary Variable Units
45	Trim Loop Current Zero. Adjusts the loop current to 4mA. Does not affect the range values or the digital primary process value
46	Trim Loop Current Gain. Adjusts the loop current to 20mA. Does not affect the range values or the digital primary process value

Table 2.5: Examples of HART Common Practice commands

Cmd	Description
128	Read Unsigned Char Variable
129	Write Unsigned Char Variable
130	Read Unsigned Int Variable
131	Write Unsigned Int Variable
132	Read Float Variable
133	Write Float Variable
134	Read String Variable
135	Write String Variable
140	Reset Totalizer and Overflow
141	Reset Error Register and Mains Interrupt Counter
150	Lese Spektrum

Table 2.6: Examples of HART Device Specific commands

2.4 IEEE 802.15.4 standard

802.15.4 is a standard for wireless communications issued by the Institute for Electrical and Electronics Engineers (IEEE). IEEE 802.15.4 is a standard which specifies the Physical layer (PHY) and Media Access Control (MAC) for Low Rate Wireless Personal Area Networks (LR-WPANs). 802.15.4 is the core for many wireless protocols such as ZigBee, International Society of Automation 100.11a (ISA100.11a), WirelessHART, and Microchip Wireless (MiWi). The upper layers are not defined in IEEE 802.15.4 standard; such layers must be defined in the protocols themselves.

2.4.1 PHY and MAC layers

At the PHY layer, the 802.15.4 standard determines that communications must occur in any of these three frequencies: 868 − 868.8MHz, 902 − 928MHz or the 2.400 − 2.4835GHz for the Industrial, Scientific and Medical (ISM) bands. Although any of these bands can be used by 802.15.4 devices, the most popular band is the 2.4GHz which is open in most of the countries worldwide with 16 channels. The 868 − 868.8MHz band is mainly used in Europe with 1 channel whereas the 902 − 928MHz band is used in the United States and Canada with 10 channels. The 5MHz channels ranging from 2.405 to 2.480GHz occur in the 802.15.4 standard, specifically the 2.4GHz band. In this band, the maximum over-the-air data rate is around 250 kbps with 10-meters communication range. Although, these theoretical data rates can be reduced in half due to overhead in the communication. Lower data rates between 20 and 40 kbps were initially

defined, however the newer versions support up to 100 kbps without any difficulty. The 802.15.4 standard at 2.4GHz specifies the use of Direct Sequence Spread Spectrum (DSSS) and uses an Offset Quadrature Phase Shift Keying (O-QPSK) with half-sine pulse shaping to modulate the RF carrier.

Geographical regions	**Europe**	**Americas**	**Wordwide**
Frequency assignment	868 – 868.8MHz	902 – 928MHz	2.400 – 2.4835GHz
Number of channels	1	10	16
Channel Bandwidth	600kHz	2MHz	5MHz
Data rate	20 kbps	40 kbps	250 kbps
Modulation	Binary Phase Shift Keying (BPSK)	BPSK	O-QPSK

Table 2.7: Comparison between IEEE 802.15.4 PHY layer according to geographical regions.

At the MAC layer, 802.15.4 uses a couple of techniques to avoid collisions when nodes start emitting at the same time on the wireless medium. Firstly, as with many wireless technologies, the MAC is carried out with the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol in order to avoid collision between devices. This method is described as follows: each node listens to the medium before transmitting, then if the medium is not available or free to transmit, the transceiver waits a random time and later tries again. Secondly, the 802.15.4 standard has a feature called real-time suitability that influences the reservation of the Guarantee Time Slots (GTS). This technique uses a centralized node which offers timeslots to each node, therefore they know when they are allowed to transmit in the medium. CSMA/CA is the most common 802.15.4 protocol in the MAC layer.

Unlike the current domestic wireless networks, 802.15.4 was taken into account for being developed with lower data rate, simply connectivity, battery application in mind and low cost. According to these constraints, the cryptographic protocols used at the upper layers are usually using algorithms of symmetric cryptography such as AES-128 in mode Counter with CBC-MAC (CCM) both for confidentiality and integrity. This will be discussed in detail later. As upper layers are dependent on the correspondent protocols, different versions of AES could be encountered.

2.4.2 Comparison of IEEE 802.15.4 standard and other wireless technologies

In Table 2.8 [53], we depict a comparison between the IEEE 802.15.4 standard with the most popular wireless technologies such as GSM/GPRS for mobile phones, IEEE 802.11 for what we understand as wireless devices (commonly known as Wi-Fi (WiFi))and Bluetooth, mainly for connectivity and pairings between small devices such as smartphones or PCs.

	802.15.4	**GSM/GPRS CDMA**	**802.11**	**Bluetooth**
Main application	Monitoring control	Voice and data	Highspeed Internet	Device connectivity
Battery life	5-10 Years	1 Week	1 Week	1 Week
Bandwidth	250 kbps	Up to 2 Mbps	Up to 54 Mbps	720 kbps
Typical range	10 - 100 meters	Various Km	50-100 meters	10-100 meters
Advantages	Low Power, cost	Existing Infrastructure	Speed	Convenience

Table 2.8: Comparison between IEEE 802.15.4 and the most well-known wireless technologies.

2.4.3 IEEE 802.15.4 Device Classes

Different devices can participate in an IEEE 802.15.4 network (Figure 2.7), such are classified depending on their functionality. The standard defines a couple of types of nodes. Therefore, we can distinguish between Full Function Device (FFD) and Reduced Function Device (RFD). An IEEE 802.15.4 network is comprised of both FFD and RFD nodes. To understand these types it is interesting to review some definitions. A coordinator is FFD with network device functionality that provides coordination between nodes and other services to the network. A Personal Area Network (PAN) coordinator is the main controller of the PAN. A network has exactly one PAN coordinator. After these definitions, let us define FFD; this kind of node can serve either as the coordinator of the PAN or as a common node in the PAN network. FFD is allowed to talk to any other device due to their total functionality. Furthermore, FFD has PAN coordinator capabilities and accepts any topology. On the other hand, RFD can only talk to a network coordinator; they cannot become a network coordinator.

2.4.4 IEEE 802.15.4 Network Topologies

In this chapter we will see the main network topologies used in the LR-WPANs. As many wireless protocols use such topologies, we describe the possibilities offered in each topology. In order to explain the most used topologies in IEEE 802.15.4, we must think about the routing organization in a WSN. The huge number of nodes necessitates a way to organize different ways of relaying data to the routers and gateways. At least a FFD working as coordinator is required in an IEEE 802.15.4 network. All devices are identified by a unique 64bit identifier, although 16bit might be used in reduced environments.

In this section, we explain the following topologies in WSNs: star and mesh topologies.

- *Mesh.* In a WSN, in this case a mesh wireless network, all nodes transmit traffic for the network. There is a cooperation between all nodes to distribute reliable data. This sort of network uses a couple of techniques for relaying traffic in the network: routing or flooding. Although routing is covered in the upper layers, the standard provides support for multi-hop communications. If routing is chosen, then the traffic is propagated along a specific route by hopping from node to node until it reaches its destination. This routing is managed by self-healing algorithms in order to detect anomalies or broken paths. In a network where all nodes cooperate, self-healing algorithms are crucial to keep the network reliable, secure and to offer redundancy. On the other hand, if flooding is used, instead of using a specific route for transmitting data from one node to another node, the data is sent to all the nodes in the mesh network, including those to whom it was not necessary. This technique is very similar to the broadcasting technique. Readers can interpret these networks as point-to-point or peer-to-peer networks. In a mesh network the scope might be larger than star topologies and sometimes a sort of cluster trees can be formed. In these structures, there is always a network coordinator which is the main coordinator in the network. Besides, there are trees comprised of at least a PAN coordinator, FFD and RFD. Since RFD are only capable of associating with one FFD at a time, the cluster tree structure seems to have RFD and non-coordinator FFD as leaves of the tree. Originally, mesh networks were invented for military intentions.

- *Star.* A star topology is the WSN in which all nodes are directly connected to a central node, creating a 'star'. It is very important that the central node is the main PAN coordinator and its nodes are FFD or RFD. All nodes transmit data to the central node, which is responsible for transmitting data from within the network to outside of it. Each node cannot directly connect with another node; it always needs to be routed through the central node. The main advantage is that all data is centralized, however a very big disadvantage is that when the central node breaks down, the WSN ceases to function properly.

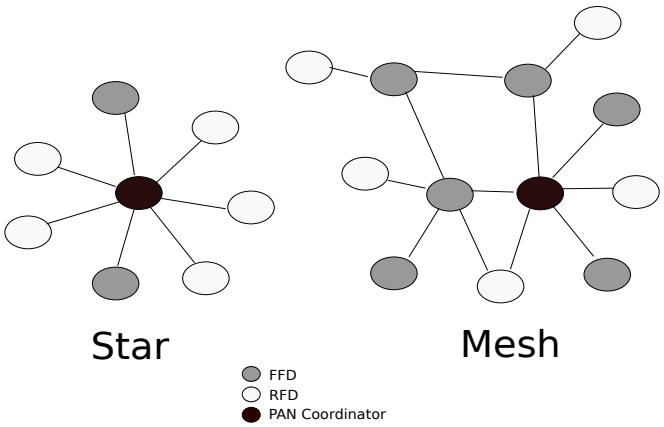


Figure 2.7: Network topologies in 802.15.4

2.4.5 IEEE 802.15.4 Packet Structure

The frame structures have been robustly designed to work on noisy channels and reduce the complexity to a minimum. Each protocol layer encapsulates its payload, adding layer-specific headers and footers to be distinguished in the upper layers [54]. The IEEE 802.15.4 standard defines four different kinds of frames structures:

- Beacon Frame. These frames are used by the PAN coordinator to transmit beacons and synchronize with other devices. Coordinators are constantly broadcasting beacon frames to another nodes to establish a communication. Beacon frames contain network information.
- Data Frame. This is used for transmitting all the data. This frame is normally called the payload.
- Acknowledgement Frame. This is used for confirming successful frame receptions.
- MAC Command Frame. This is responsible for handling all MAC operations. These MAC frames are mainly commands to indicate operations such as: association request and response, disassociation notification, data request between many others.

Each packet or Physical (PHY) Protocol Data Unit (PPDU) can be comprised of the following parts: Synchronization Header (SHR), PHY Header (PHR) and a PHY Service Data Unit (PSDU). As Figure 2.8 depicts, such headers and Data units are encapsulated in different necessary fields to fulfil the IEEE 802.15.4 standard. On the other hand, the PSDU at the PHY layer is itemized in detail at the upper layer. As Figure 2.8 also shows, a PSDU is constituted by a MAC Header (MHR), MAC Service Data Unit (MSDU) and MAC Footer (MFR). In the MHR, addressing fields are filled up with both the source and destination addresses to redirect the payload. The rest of these fields are required to correctly follow the standard.

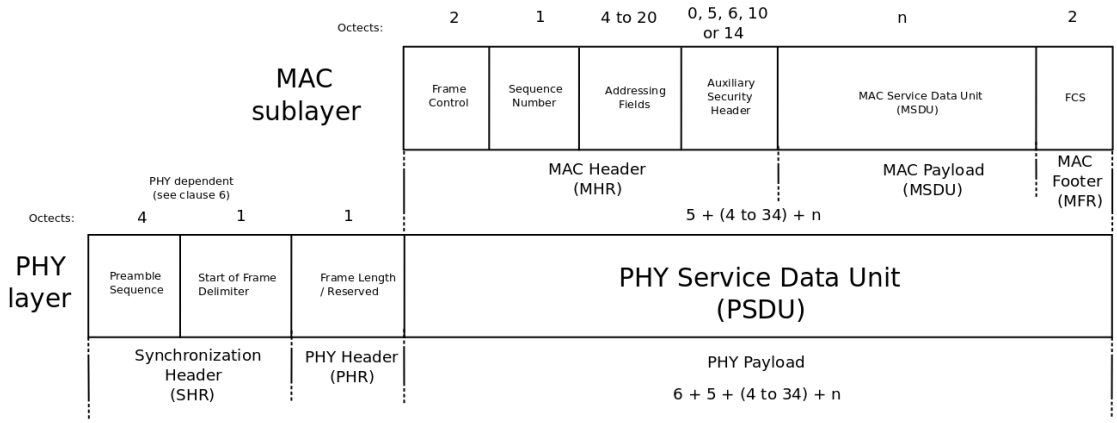


Figure 2.8: Packet structure in PHY and MAC layers for the IEEE 802.15.4 standard.

2.5 WirelessHART

With more than 30 million HART devices over the entire world, HART technology is most widely used field communication protocol in industrial environments. On account of this, the industrial community has not slowed down of attempting to reduce costs. Million of devices' wires could be removed by employing wireless technology. Researchers and the industrial community have invested time and effort in order to develop a secure and efficient new wireless protocol for the industrial environment. Despite of reducing costs, wireless technologies are also well-known for their ease in deployments. WSNs have strongly emerged in industrial environments as well as many other fields.

WirelessHART is a wireless communication protocol for process automation applications based on the proven and widely-used wired HART protocol. Summarised WirelessHART, the HART protocol enhanced with wireless capabilities. Nonetheless, the HART compatibility between WirelessHART and HART devices is always maintained. According to its website [12], WirelessHART claims to consistently fulfil all specific requirements for reliability, security, cost-efficiency and ease of use. Another claim coming from its website: "It is a simple, reliable and secure protocol!".

2.5.1 Main Characteristics

This section intends to introduce the main aspects for the WirelessHART protocol. WirelessHART is the first open wireless standard to be certified for industrial applications. Being an open system allows WirelessHART to be a multi-vendor protocol and worldwide usable. Wireless networks can be set up with different brands and all devices are capable of understanding each other.

In plant environments the communications are managed by using frequency hopping. In WirelessHART there are 16 channels as discussed before in the IEEE 802.15.4 standard (Section 2.4). Each channel has a bandwidth of 5 Mhz and their allocation occurs between 2.405GHz and 2.480GHz. The WirelessHART protocol uses all channels in parallel to avoid overlaps. If a channel is being used, then the Network Manager can utilize frequency hopping to change to another open channel and continue with it without interruption. WirelessHART employs Time Division Multiple Access (TDMA) to manage how the spectrum is used over time. Every transmission takes place in a 10ms window called a "time slot" and on one of the 16 channels with very precise timing. If no communication is needed, devices go into a sleep mode to conserve energy (most devices are battery powered). Therefore, the whole WirelessHART is synchronized in these

[12] http://en.hartcomm.org/hcp/tech/wihart/wireless_overview.html

slots and the rates can reach up to 250kbps.

WirelessHART was implemented with the idea of keeping the well-known and proven HART protocol and creating it as easy as using wired HART. The first aspect that differs from the HART protocol is obviously the RF interface. WirelessHART is complying with the IEEE 802.15.4 – 2006 [13]. As we mentioned in the previous section 2.4, IEEE 802.15.4 is responsible for the PHY and MAC layers for LR-WPANs used WirelessHART as well. The HART protocol uses a modulated sine on the 4 – 20mA current loop, whereas in WirelessHART this is replaced by radio. This feature produces costs savings which make the WirelessHART an attractive protocol besides its easy and flexible installation.

2.5.2 WirelessHART Components

According to the standard, a WirelessHART network architecture is constituted by different elements. Figure 2.9 depicts all the device classes cited in the WirelessHART standard. Below such elements are discussed and briefly explained:

- *Field Devices.* The wireless sensor nodes are also called motes and they act as routers. The WirelessHART protocol normally supports up to 250 nodes in the same network although some devices are capable of reaching up to 500 nodes. Such nodes carry out different objectives in the WSN as gathering and preprocessing environmental information and communicate with other nodes in the same network. Finally, this information is supposed to arrive to the Access Point (AP)s where nodes are connected to. This information is going through the gateway until it reaches the final point of monitoring.

- *Adapters.* These provide wireless capabilities to the wired HART devices.

- *AP.* A wireless AP is responsible of transmitting all the wireless data to the Gateway. Although there can exist WirelessHART networks without APs, deployed them into the network can reduce the workload for the gateway, specially huge WirelessHART networks. Such devices are wirelessly transmitting and receiving data from the wireless network and relaying to the core network by using wired connection with the Gateway. Furthermore, both AP and Gateway can communicate with each other through the air.

- *Security Manager (SM).* The Security Manager is responsible for creating and storing the keys used in the network. The Network Manager uses the Security Manager for key management. Besides, the Security Manager can also be used to ensure the wired communications of the WirelessHART network.

- *Network Manager (NM).* The NM is a centralized entity responsible for configuring and scheduling the WSN. The NM accepts joining requests from the Gateway, AP, field devices, adapters and handheld devices [55]. The NM can be integrated into the Gateway (GW), host application or process automation controller.

- *GW.* A GW is an AP responsible for joining the plant automation network and the wireless network. It is the link between the NM and the WirelessHART network.

- *Handheld Terminal.* Portable devices used to configure the field devices. Used mostly to write the Network ID or Join key and device monitoring.

- *Host Application.* This is the machine that is monitoring constantly.

[13] http://standards.ieee.org/findstds/standard/802.15.4-2006.html

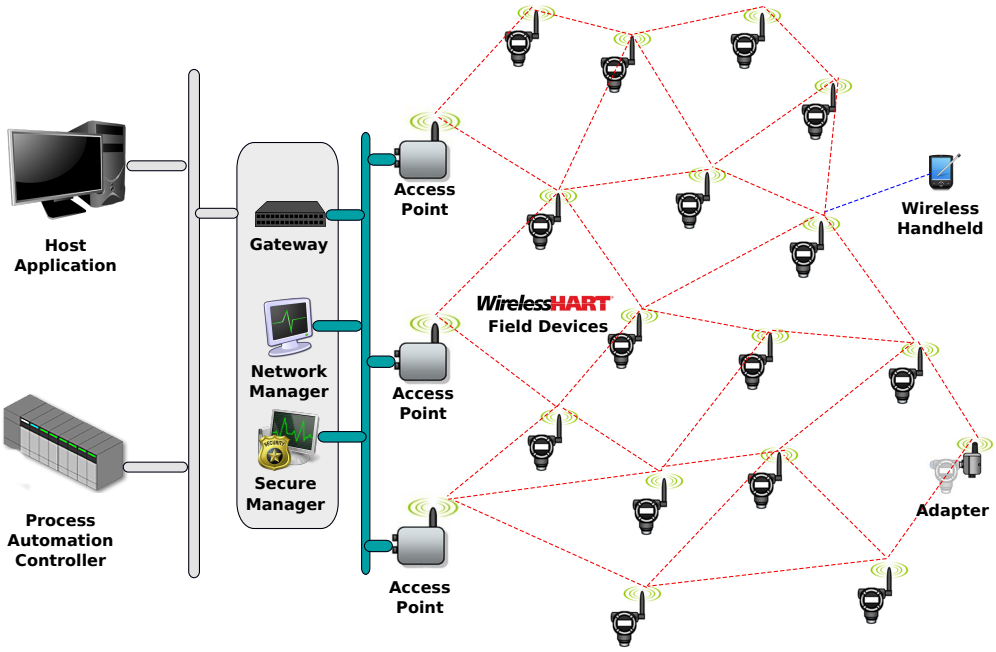


Figure 2.9: WirelessHART Mesh Network Architecture

2.5.3 WirelessHART Communication Layers

This section intends to provide a global view of the WirelessHART packet structure and its security. In Figure 2.10, a HART packet structure is depicted at different layers whereas Figure 2.11 shows the WirelessHART packet structure. The number attached in each field represents the number of octets or bytes used and the colours help to have a quick interpretation of layers. Eventually, in Figure 2.13 addresses a security comparison between the 7 layers of the OSI model in WirelessHART and HART communication layers. In addition to that, we provide a brief detail about the main features of each layer improving an existent comparison [1].

As Figure 2.12 depicts, a dissection of a WirelessHART packet at different layers from the PHY layer until the application layer. Both the Data Link Layer (DLL) and the network layer contain security measures. At the DLL the DLL Protocol Data Unit (DLPDU) is authenticated through a Message Integrity Code (MIC) using AES-128-Cipher Block Chaining Message Authentication Code (CBC-MAC) encryption, whereas at the network layer the Network Protocol Data Unit (NPDU) is both authenticated by a MIC and encrypted using AES-128-Counter (CTR) mode. At the upper layers, transport and application layers, the data is neither authenticated nor encrypted. However, these layers wrap the encrypted and authenticated data previously at the lower layers.

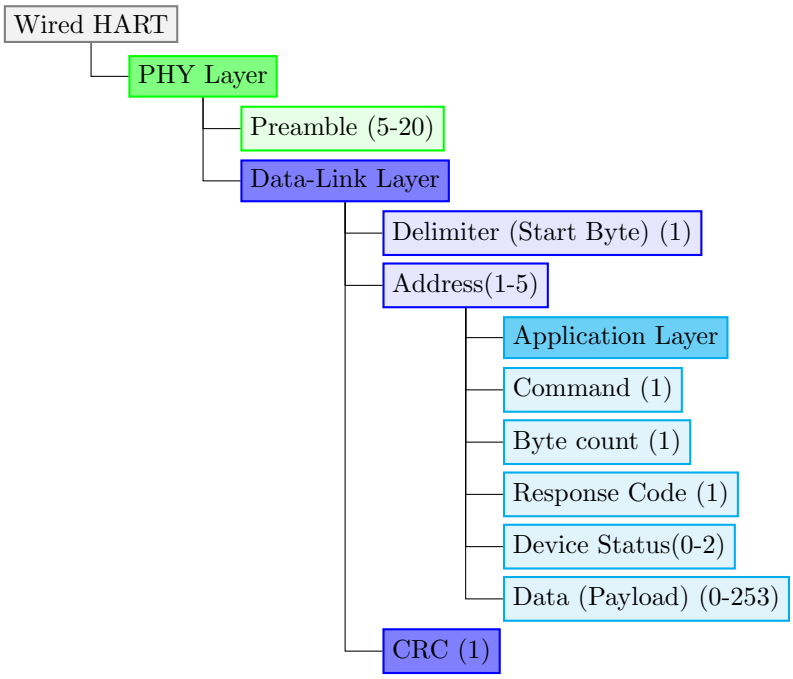


Figure 2.10: Packet structure in HART.

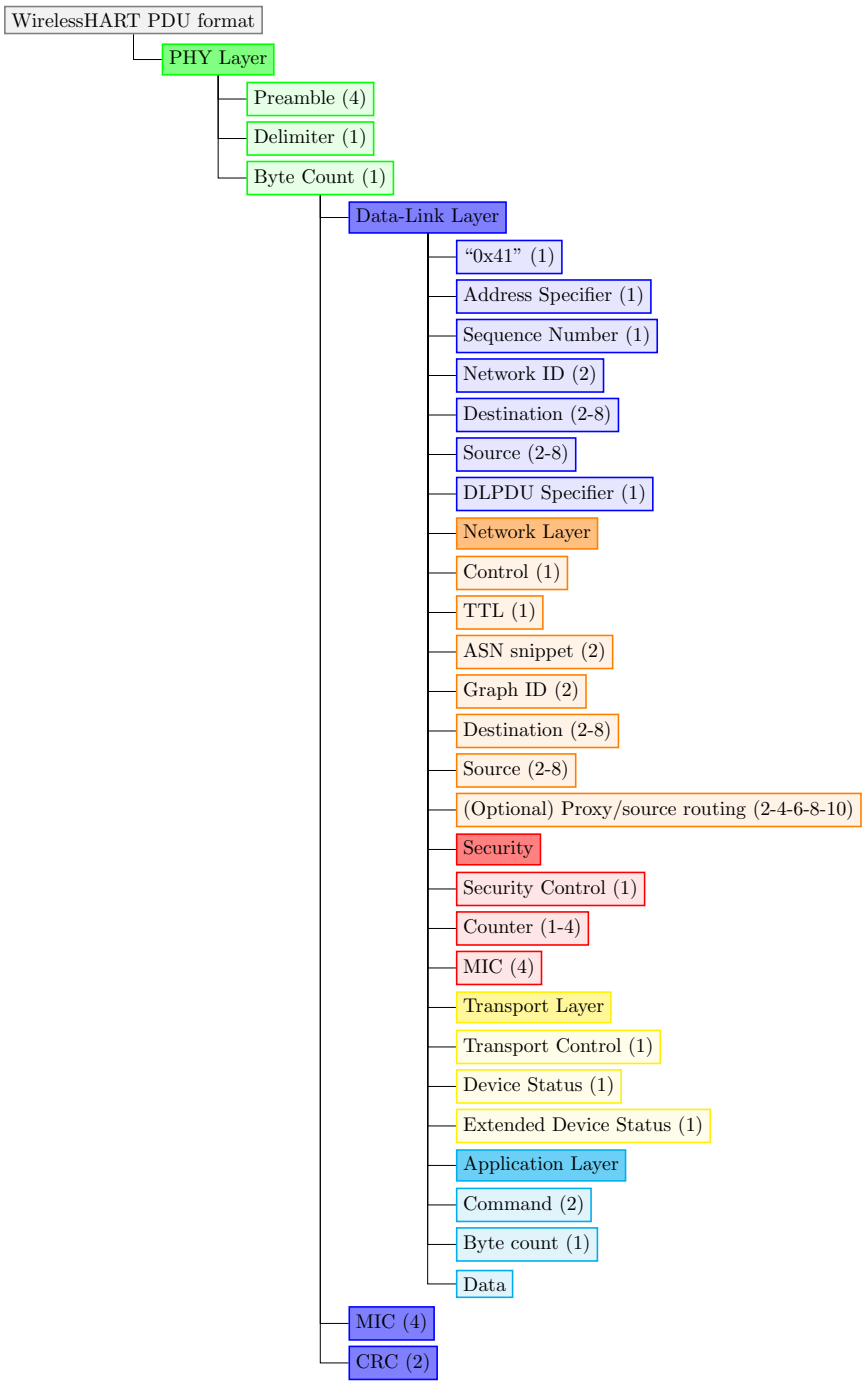


Figure 2.11: Packet structure in WirelessHART.

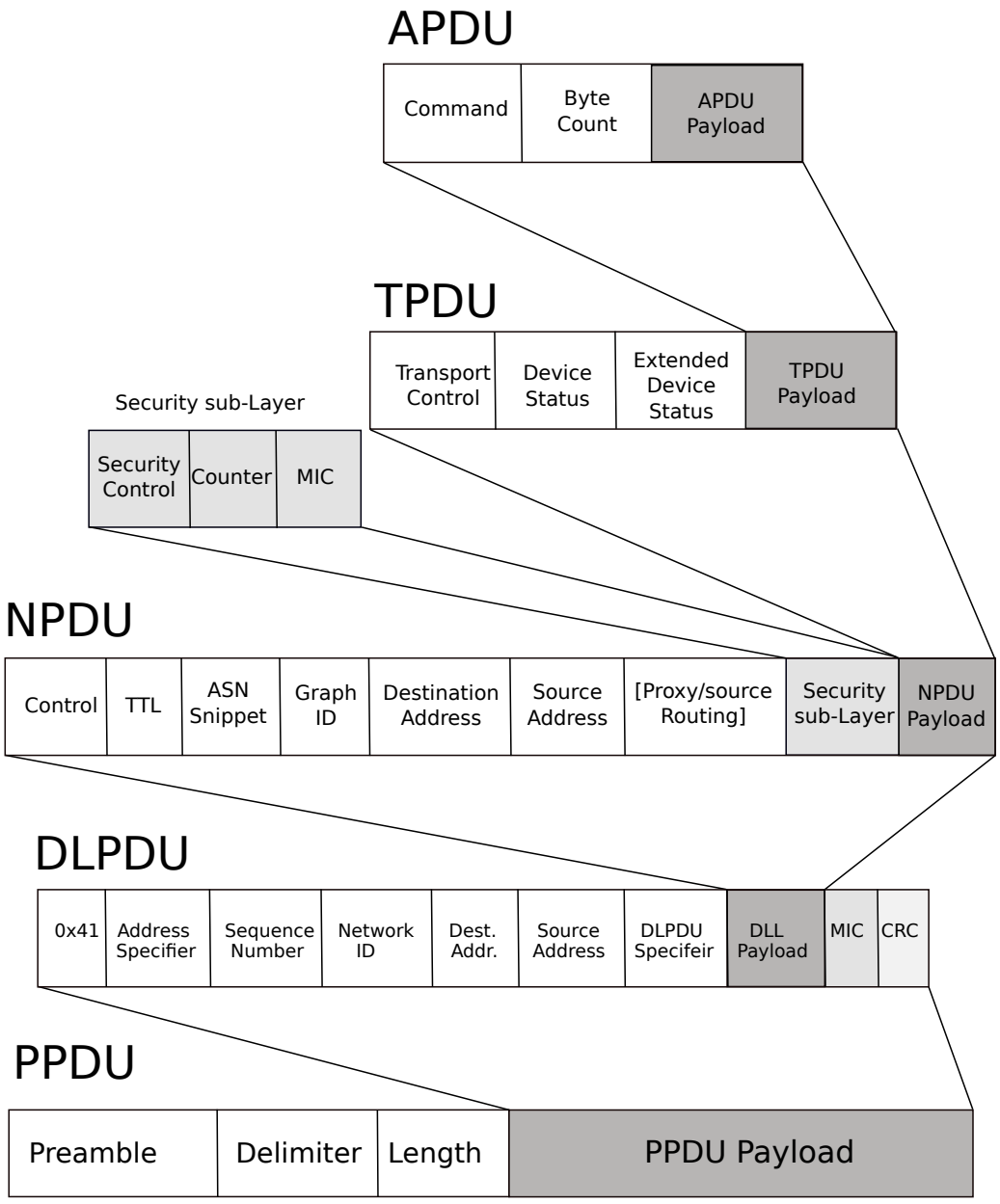


Figure 2.12: Dissection of a WirelessHART packet at different layers.

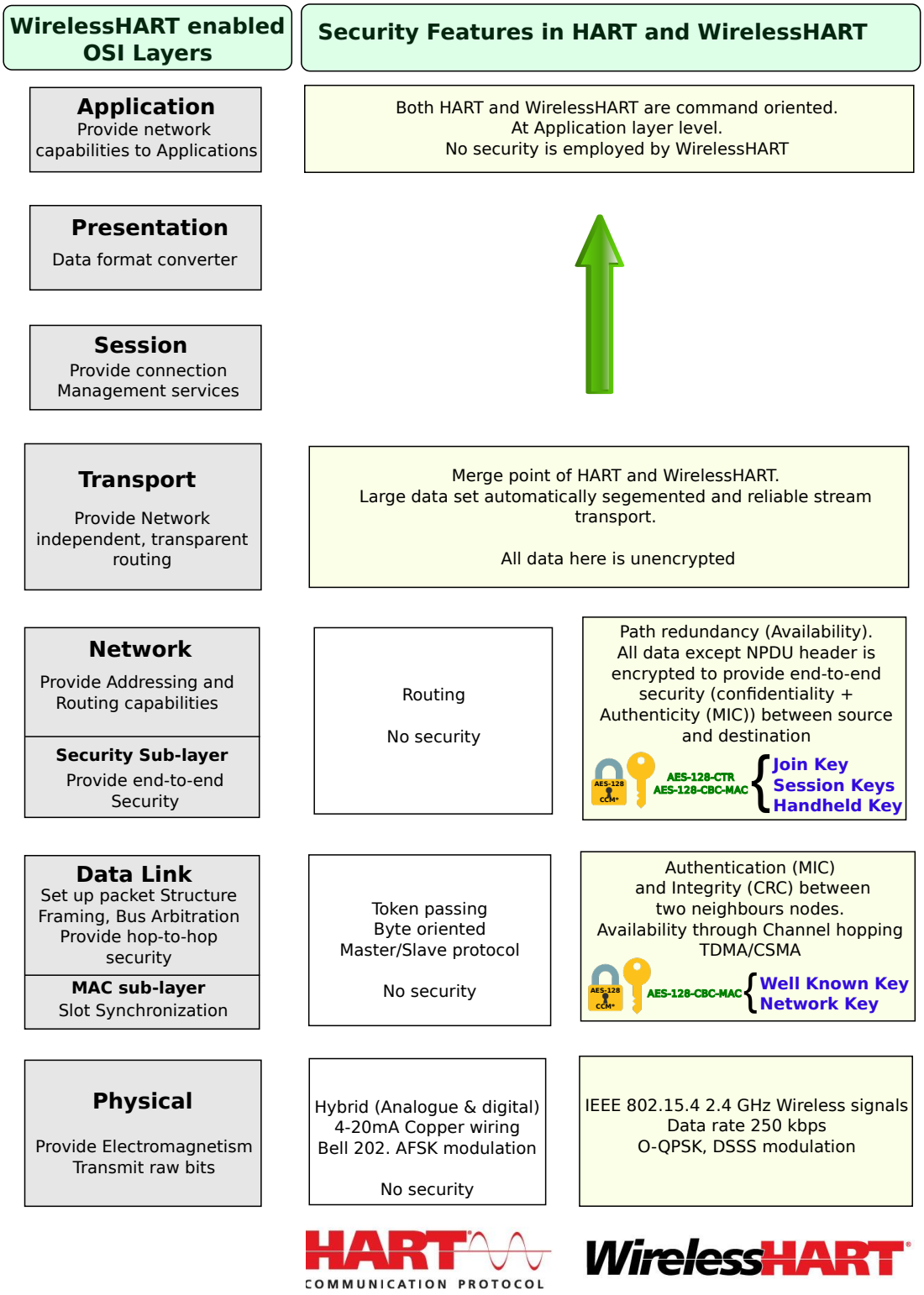


Figure 2.13: Security features comparison between HART and WirelessHART. ([1] improved)

2.6 Cryptography in WirelessHART

WirelessHART inherits features of the mesh networking technology but at the same time it also incorporates many features of WSNs. The main goal in sensor networks is to monitor the actual process and obtain as much reliable data as possible. If an adversary is able to manipulate the network offering fake data, then our task would be compromised and tainted. Side effects could be very disastrous producing unexpected situations. As solution of that, WirelessHART designers ensured a secure communication between all the nodes with each other. All wireless data in WirelessHART fulfils confidentiality, integrity, and the authenticity of the commands is ensured. WirelessHART provides security just for the wireless part of the network but neither enforces to protect nor specifies any solution for the wired part of the network [1].

In this section, we explain how WirelessHART achieves confidentiality, integrity and authenticity in the wireless network. WirelessHART was designed with security in mind and all the WirelessHART data is authenticated and encrypted on each layer using symmetric cryptography, specifically AES-128 in CCM* mode. Nowadays, AES-128 bits is considered secure enough and the combination of two well-known modes of AES provides both authentication and encryption of all wireless data at the same time.

2.6.1 Outline

This section briefly discusses the AES cipher and explains a bit further the most used modes in WSNs, especially in WirelessHART. This section is neither an in-depth discussion about how AES was implemented nor all the AES modes available nowadays. Nonetheless, we attempt to dissect the most important details about AES in WSNs and WirelessHART. We shortly mention the main aspects of AES and we pinpoint what it is relevant for wireless networks as well as comment the CCM* mode used in WirelessHART.

2.6.2 AES

Around 1997, the U.S. National Institute of Standards and Technology (NIST) announced their wish to have a new encryption standard which would substitute the Data Encryption Standard (DES). The winner of a crypto-contest organized by NIST was the *Rijndael* cipher that was developed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen. AES is a block cipher of 128 bits with three different key lengths of 128, 192 and 256 bits. Unlike its predecessor DES, AES is not a Feistel network but is a substitution-permutation network. AES uses a $4x4$ matrix structure known as AES state which can be interpreted as operations in the Galois Field (GF) (2^8). As also DES carried out, the AES cipher is based on rounds. Such rounds are repeated a number of times. In these rounds, different AES keys are created from a key schedule. This function takes as input the AES key and generates the keys. AES has 10 rounds for 128 bit keys, 12 rounds for 192 bit keys and 14 rounds for 256 bit keys. During the so-called rounds, the AES state is processed through different operations:

- *SubBytes.* This function substitutes each byte with another byte according to a lookup table.
- *ShiftRows.* This function is shifting the last three rows of the state cyclically a certain number of steps.
- *MixColumn.* This operation mixes up the AES state columns using an irreversible linear transformation.
- *AddRoundKey.* This operation combines each byte of the AES state with a block of the round key by using bitwise XOR.

Apart from the above mentioned operations, AES can operate in several modes and there are six confidentiality modes: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feed Back (CFB), Output Feed Back (OFB), CTR and XTS-AES. The CBC-MAC and Cipher-based MAC (CMAC) are similar authentication modes only. Furthermore, there are five combined modes for confidentiality and authentication: CCM, GCM, KW, KWP and TKW. Only modes used in WirelessHART are discussed in this thesis.

2.6.3 AES CTR

The AES-CTR mode was introduced by Diffie and Hellman already in 1975. The main idea of this mode consists of a counter, a unique per-packet value. The AES-CTR uses this counter as input for the AES encryption operation with the round key. This counter might be considered as an Initialization Vector (IV) which is a 128-bit fresh number and the same combination of IV and key should never be used more than once. The counter structure is as follows: a 32-bit nonce, 64-bit IV and the least significant 32-bit a counter starting at one [56]. The nonce as its name claims is a single use value. This nonce has to be assigned at the beginning of the security association and it needs to be unpredictable. The IV is chosen by the encryptor and it ensures no encryptions with the same IV-key pair. The leftmost 32-bits are a simple counter starting at zero or one depending on the application and it is incremented subsequently in order to generate the keystream with the round key. After the counter has been encrypted with the round key, the resultant key stream is combined with the plaintext or message by performing a bitwise XOR producing the first encrypted block as final output.

The AES-CTR mode is well-known for being fast and easy to pipeline. Different techniques have been applied to this mode to make it as fast as possible depending on the architecture of the machine. CTR mode is thoroughly considered parallelizable and many blocks can be carried out at the same time both in hardware and software implementations. Disadvantages are the easiness of error propagation detection that can help adversaries to attack the encryption [57]. On top of that, this mode only offers confidentiality and a disadvantage is the lack of data integrity.

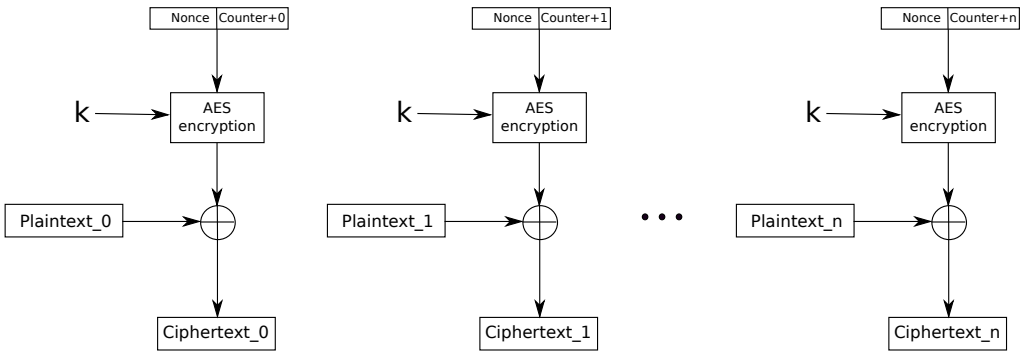


Figure 2.14: AES-CTR encryption mode

2.6.4 AES CBC

Before discussing AES CBC-MAC authentication mode, it is appropriate to explain the basics of the AES CBC encryption mode. IBM invented the CBC mode in 1976. Before encrypting this mode combines each block of plaintext with the previous ciphertext block by performing a bitwise XOR. In this way, each block is dependent on its predecessor. In order to create unique messages, an IV does need to be added in the first block because there is no previous encrypted block to be combined with.

In the AES CBC mode the main disadvantage is that it cannot parallelize the encryption. However, the decryption can be carried out in a parallel mode. Furthermore, a simple bit flipped would affect the entire subsequent blocks and would produce no clues to apply cryptoanalysis. Unlike AES ECB, CBC offers different encryptions with the same plaintext thanks to the IV. In ECB mode the same input produces the same output generating unnecessary leakages.

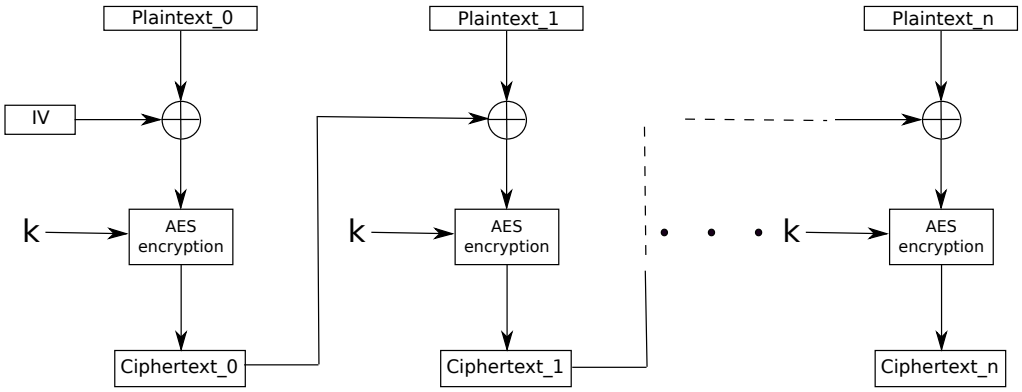


Figure 2.15: AES-CBC encryption mode

2.6.5 AES CBC-MAC

The AES CBC-MAC is an authentication mode where a MIC is generated also known as Message Authentication Code (MAC) from a block cipher. Since this mode is using the CBC mode, a slightly change in one bit produces huge changes in the output. In this mode, the message is combined with a bitwise XOR operation with the previous encryption and starting with an IV normally fixed to zero. The output of this operation is encrypted with the round key and subsequently this key stream is the input for the next bitwise XOR operation instead the IV. In this mode all data is discarded except the final block which outputs the Message Authentication Code (MAC). The MAC can be either a 32, 64 or 128 bits long. In WSNs, especially in Zigbee and WirelessHART this MAC is set up to 32-bits (4 bytes). This mode can be used for both plaintext and ciphertext. An important aspect to keep in mind is that this mode needs the exact number of blocks and padding can be applied whether there is not data enough to calculate with.

AES CMAC is a variant of AES CBC-MAC where variable lengths of messages are acceptable without losing security. The AES CMAC addresses the security deficiencies of the AES CBC-MAC. See [58] and [59] for further information.

2.6.6 AES CCM*

Counter with CBC-MAC (CCM) is a generic authenticated encryption block cipher mode. This mode combines the use of AES-128 in CTR mode for encryption and generates the MIC with CBC-MAC mode. Both CTR and CBC modes ensure high-level security that includes both data integrity and encryption at the same time. The AES-CTR mode is employed for the encryption and decryption of WirelessHART NPDU payload. On the other side, the AES-CBC-MAC mode is used for signing both at the network layer and the DLL with different keys. The difference between AES-CCM and AES-CCM* is principally that whereas AES-CCM is a combination of both encryption and authentication, AES-CCM* can offer either just encryption or authentication [60].

AES-CCM* needs 4 byte-strings as parameters: *a, m, N, K*. Where *m* is the message to be enciphered, *K* is the 128-bit AES key, *N* is the 13-byte Nonce used to avoid replay attacks and

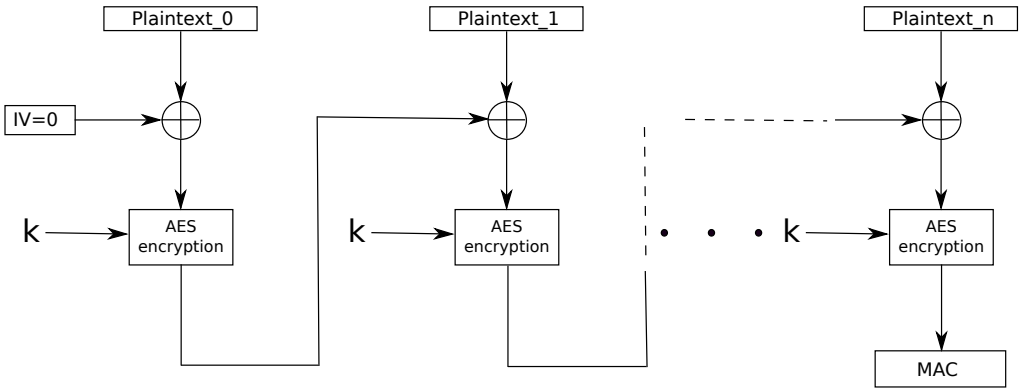


Figure 2.16: AES-CBC-MAC authentication mode. Calculation of the MIC or MAC.

a is the additional data to be authenticated but not enciphered. The latter is covering the byte delimiter until the DLPDU payload.

2.7 Key Management in WirelessHART

Key management is undoubtedly one of the most important aspects regarding the security in the WirelessHART standard. Public Key Infrastructure (PKI) is not used due to power-limited devices or at least the standard states this statement. Possibly future implementations could use PKI cryptography to improve the security in this industrial protocol. WirelessHART networks are comprised of wireless and wired devices. However, the wired security is not defined nor mandatory. Some weaknesses are found; neither non-repudiation nor security at the wired side are achieved in the WirelessHART [1]. This standard only secures the wireless part using symmetric cryptography.

In WirelessHART there is a trusted authority that is responsible for generating, storing, revoking and distributing cryptographic keys. This trusted authority is the SM and it is combined with the use of a Network Manager that distributes these keys over the network. The WirelessHART standard defines the amount of keys needed and their usage but does not specify how keys must be generated, stored, revoked, renewed and distributed. It is well-claimed that the SM is responsible for such functions but neither the design nor specific functionalities are declared in the standard. Besides, the communication between the SM and NM is poorly documented. At this moment, we know why and which keys are needed but we do not fully understand all the steps. Often the SM and NM are integrated in the Gateway, but this is not always the case.

2.7.1 Cryptographic Keys in WirelessHART

In this section we clarify the keys used in the WirelessHART communications. First of all, we discuss some aspects of wireless keys and then offer some diagrams to help readers to thoroughly understand the key management and its distribution. In order to complete this section, the following papers were useful: [55], [1] and [61]. The WirelessHART standard defines the following wireless keys:

- *Well Known Key (WKK).* This key is a hardcoded key in the WirelessHART standard and it is always the same. According to the standard, the WKK was *randomly chosen* and is: **7777 772E 6861 7274 636F 6D6D 2E6F 7267**. Apparently this is not randomly chosen, the WKK is the ASCII translation of : "`www.hartcomm.org`". This key is only used during the joining process for calculating a MIC. This MIC is attached at the DLL for either joining requests and responses or advertisements. As the key is widespread and known to everyone, it is only used for authenticating beacons and calculating MICs for joining requests and

responses. This MIC is used corresponding to the DLPDU's MIC previously addressed in Figure 2.12.

- *JK*. This is considered the main key of the WirelessHART standard. The JK is a secret and PSK used to be authenticated with the NM. The most common scenario relies on all devices having the same key which is installed manually through the maintenance port before being deployed. As Access Control List (ACL)s allow for different join keys per device. The JK is used only during the joining process and both the join request and response are encrypted with such key at the network layer. At this layer, the JK is basically used for encrypting the payload and calculating another MIC at the network layer. This MIC is corresponding to the NPDU's MIC previously addressed in Figure 2.12.

- *Session Keys*. Once the joining is started, the SM will create different session keys. These run-time keys are employed after the joining process at the network layer. Each device knows the fresh keys to encipher with during the session. Such keys are:

  - *Unicast Network Manager Key (UNMK)*. This key is ensuring end-to-end encryption between NM and the authenticated wireless nodes in the network. This session key is used for device management such as asking for the network's health, timeslots and further information. It is employed for renewing the JK whenever necessary.
  - *Unicast Gateway Key (UGWK)*. This key is ensuring end-to-end encryption between GW and the authenticated wireless nodes in the network. This key is used for securing the NPDU payload and the MIC calculation at the network layer.
  - *Broadcast Network Manager Key (BNMK)*. This key is used for sending global secure messages into the wireless network between the NM and the wireless devices (inclusive the GW). For instance operations such as routing information or network scheduling amongst others.
  - *Broadcast Gateway Key (BGWK)*. This key is used for sending global secure messages into the wireless network between the GW and the wireless nodes (exclusive the GW). For instance operations such as notifications or timings amongst others.

- *Network Key (NK)*. The NK is generated by the SM and later on this key is distributed to all the authenticated devices in the WSN by the NM. This key is responsible for calculating keyed MIC to secure the DLPDU. It is used after the joining process in order to substitute the WKK. This key is used by two nodes to authenticate each other at the DLL by verifying the MIC calculated using AES-128-CBC-MAC. Furthermore, the NM uses this key for renewing the broadcast session keys. The NK is also called "link key" in some papers.

- *Hand-held Key (HK)*. This key is used for peer-to-peer wireless communications between the handheld and the field device without passing over the GW. As discussed before in the HART section 2.3, the Handheld devices are able to connect a field device using an AFSK modem. This key is used for securing the NPDU.

2.7.2 Joining Process

Mote joining is an operation that occurs when an unauthenticated mote wants to join to the mesh WirelessHART network. A mote needs to deal with a security handshake in order to be connected to the network. This security handshake is a communication between an unconnected mote and the network manager (NM). Although the NM is the responsible for distributing all the keys during the WirelessHART joining process, the SM is the secure interface that creates and stores all keys. Therefore, several interfaces are involved in the joining process. It is known that in WirelessHART when various nodes want to speak each other, they have to go through the GW.

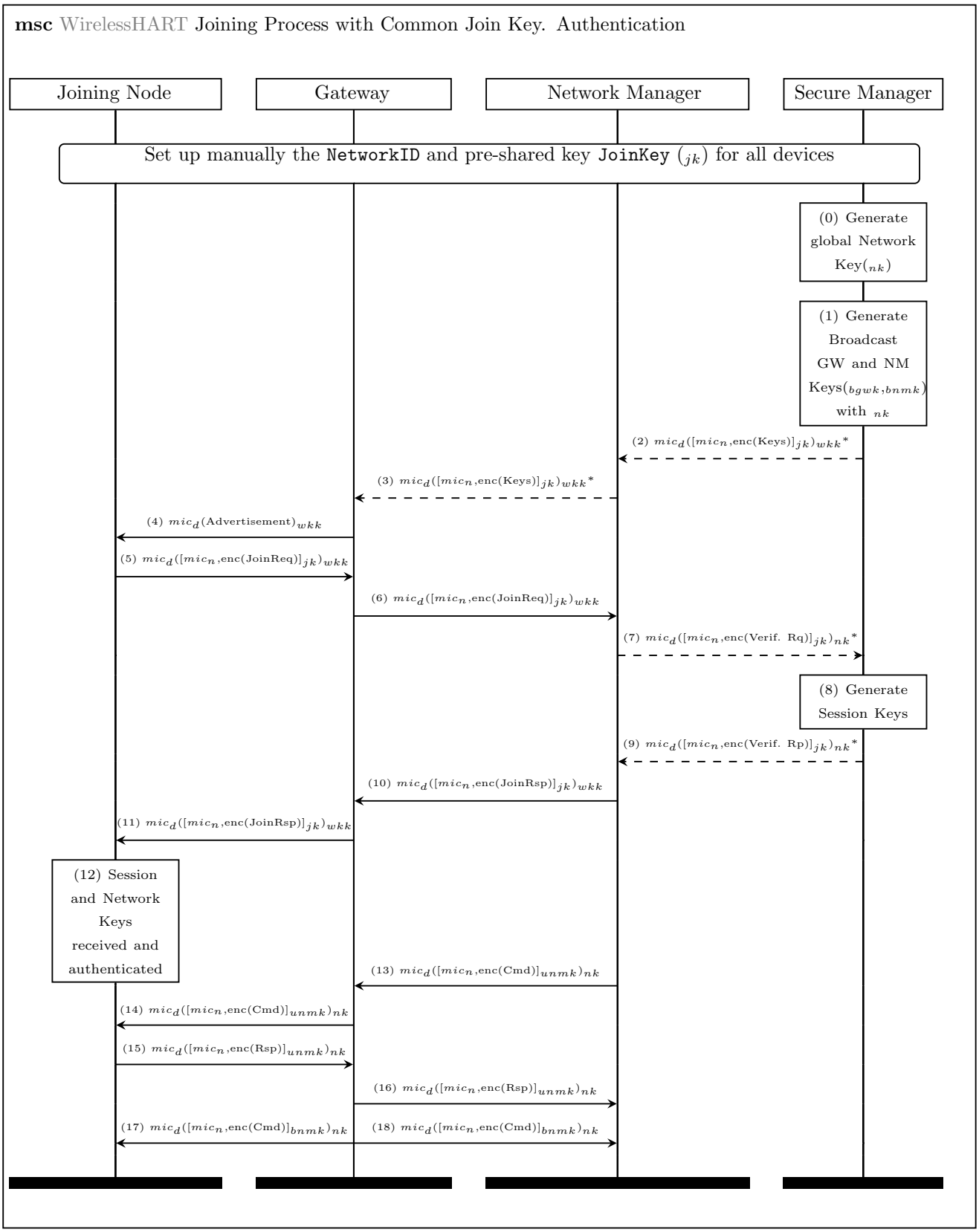
In the Joinkey establishment, the trusted administrator can configure the join key to be either a single key for all devices in the network or unique per device. The common join key is

only visible through the gateway interface. Users can recover the JoinKey by visiting the gateway web interface or accessing the system with administrator privileges to see the Joinkey. Only the administrator can change the Joinkey(s). The common (or each individual) join key can be changed in the WirelessHART gateway at any time. This change is securely propagated through the WirelessHART mesh network; old join keys become obsolete. According the joining time, this is partly determined by the join duty cycle. The join duty cycle determines the radio of active listen time to doze time during the period when the mote is searching for the network [14]. The higher the join duty cycle is, the faster the joining time, with the downside of higher power consumption.

This section clarifies the joining process and its subsequent session keys used in the WirelessHART key management using protocol diagrams. A similar approach [15] has been used to improve and provide further information as well as clarifications on the process. In the diagrams, the steps are numbered and subsequently explained. In every subsection the `numbers` used are matching the drawn numbers in the diagrams. Steps remarked with a `*` or/and a `dashed line (--)` are not well-defined and they would need to be investigated further to verify them. Since the WirelessHART provides hop-to-hop security between neighbours and end-to-end security between nodes, this thesis represents that thus: $mic_d([mic_n(x), enc(payload)]ugwk)nk$. To offer hop-to-hop security at the DLL all the nodes in the network are able to calculate the $mic_d([X]ugwk)nk$ with the global network key and prove that the message is legitimate. At the network layer the same message is encrypted and signed another time and provides end-to-end security. At the network layer it is specified as $[X]ugwk$ or $[mic_n(x), enc(payload)]ugwk$. This provides data authenticity and confidentiality generating the MIC and encrypting the payload with the UGWK.

[14] http://cds.linear.com/docs/en/design-note/SmartMesh_WirelessHART_Mote_Serial_API_Guide.pdf
[15] http://www.ietf.org/mail-archive/web/6tisch-security/current/pdfiZJLN6CKg0.pdf

Joining Process with Common Join Key



The WirelessHART Joining Process with Common Join Key is depicted as follows:

0. All the devices or interfaces are set up with the same JK manually. The SM creates the global NK using the JK. The NK will provide hop-to-hop security by calculating a MIC at the DLL posteriorly.
1. The SM also creates the broadcast keys using the NK.
2. The SM sends the session keys to the NM. This step is not in the WirelessHART standard and it might have something uncompleted.
3. The NM forwards the session keys towards the GW. Likewise step 2 and it might have something uncompleted.
4. The joining process starts with advertising. All the nodes start beaconing the spectrum to offer the possibility to join the network for the unauthenticated nodes. These advertisements are signed with the WKK using AES-128-CCM*. In the diagram this appears as $MIC_d()$ and it is carried out at the DLL.
5. The unauthenticated mote is awake and receives the advertisement. The mote recognises the Network ID and starts sending a Join Request. This request is enciphered with the JK and authenticated too at the network layer and authenticated with the WKK at the DLL.
6. The GW forwards the Join Request to the NM verifying the MIC with the WKK.
7. The NM needs to verify this Join Request and asks the SM to generate the run-time session keys for this new mote.
8. The SM creates the session keys. The way of generation those keys are generated is unknown although hashing techniques are used in some public implementations [61].
9. The SM sends the session keys to the NM. The NM knows the keys which will be used later on.
10. The NM responds and sends the Join Request to the GW.
11. The GW forwards the Join Response to the unauthenticated mote.
12. The joining node proves the data integrity at the DLL with the WKK and decrypts the Join Request with the JK. Inside there is another proof of authenticity and the payload carrying the session keys and the NK. The joining node now knows that it is authenticated in the WSN. The joining process is stopped right here and the security handshake is considered over as well.
13. The NM needs to send some additional data to the new joined mote. The NM sends a *Command* enciphered with the UNMK and authenticated with the global NK.
14. The GW forwards the command to the destination. The mote is able to decrypt this data because it is using the unicast key (UNMK). With this end-to-end security is claimed.
15. Response back.
16. Response forwarded to the destination. The NM decrypts and gets the data in the response.
17. The GW sends a broadcast packet towards its neighbours by enciphering with BGWK and signing with the global NK. All the authenticated devices must already have the broadcast keys.
18. Likewise step 17. Broadcast.

Joining Process with Access Control List (ACL)

This modality is claimed to be more secure than the security handshake with only the common Join Key. It is known that a couple of modes can occur with the ACL. Either each node has a unique Join Key and the Manager owns an ACL with all `IDentifier-JoinKey` [16] pairs or, all nodes have the same common Join Key and the Manager has an ACL of allowed devices. The most secure mode is using unique keys for each node, although the main disadvantage of this approach is focussed on the initial key establishment. This approach is also well-known in the networking area as *MAC filtering*. Although it does not help much to strengthen the security, at least it increases the difficulty for an adversary.

The fact that a prior configuration has to be initialized both the Join Keys and the identifier can severely improve the system security. The main advantage of this approach is the non-necessity to reschedule a global joining process when a node is compromised or lost. Eventually, this mode can save lifetime for nodes and increases the security level of the WSN. However, the key establishment requires a high workload for the deployment. There exists a RAM limitation for ACL entries up to 32 motes or 100 (by using external RAM). Therefore, this scheme is infeasible for bigger than 32-nodes networks in the Linear and Dust Networks devices.

The WirelessHART Joining Process with the ACL is depicted as follows:

0. All the devices or interfaces are set up with the unique JK manually. The GW has an ACL with device_ID-Joinkey$_{id}$. Broadcast and network keys generation is assumed.

1. The joining process starts with the advertising. All the nodes start beaconing the spectrum to offer the possibility to join the network for the unauthenticated nodes. These advertisements are signed with the WKK using AES-128-CCM*. In the diagram this appears as MIC$_d$() and is carried out at the DLL.

2. The unauthenticated mote is awake and receives the advertisement. The mote recognises the Network ID and starts sending a Join Request. This request is enciphered with the unique JK and authenticated too at the network layer and authenticated with the WKK at the DLL. The Identifier travels in plaintext although signed with the JK.

3. An authenticated node acts as a proxy and forwards the Join Request to the GW

4. The GW checks if the Identifier exists in its ACL. If so, it decrypts the Join Request with the mote's unique JK which is known by the GW as well because it is in the list. Otherwise, it does not decrypt the Join Request and replies with Identifier not found.

5. The Identifier was in the ACL, so the SM generates the run-time session keys (*sk1*) for this node.

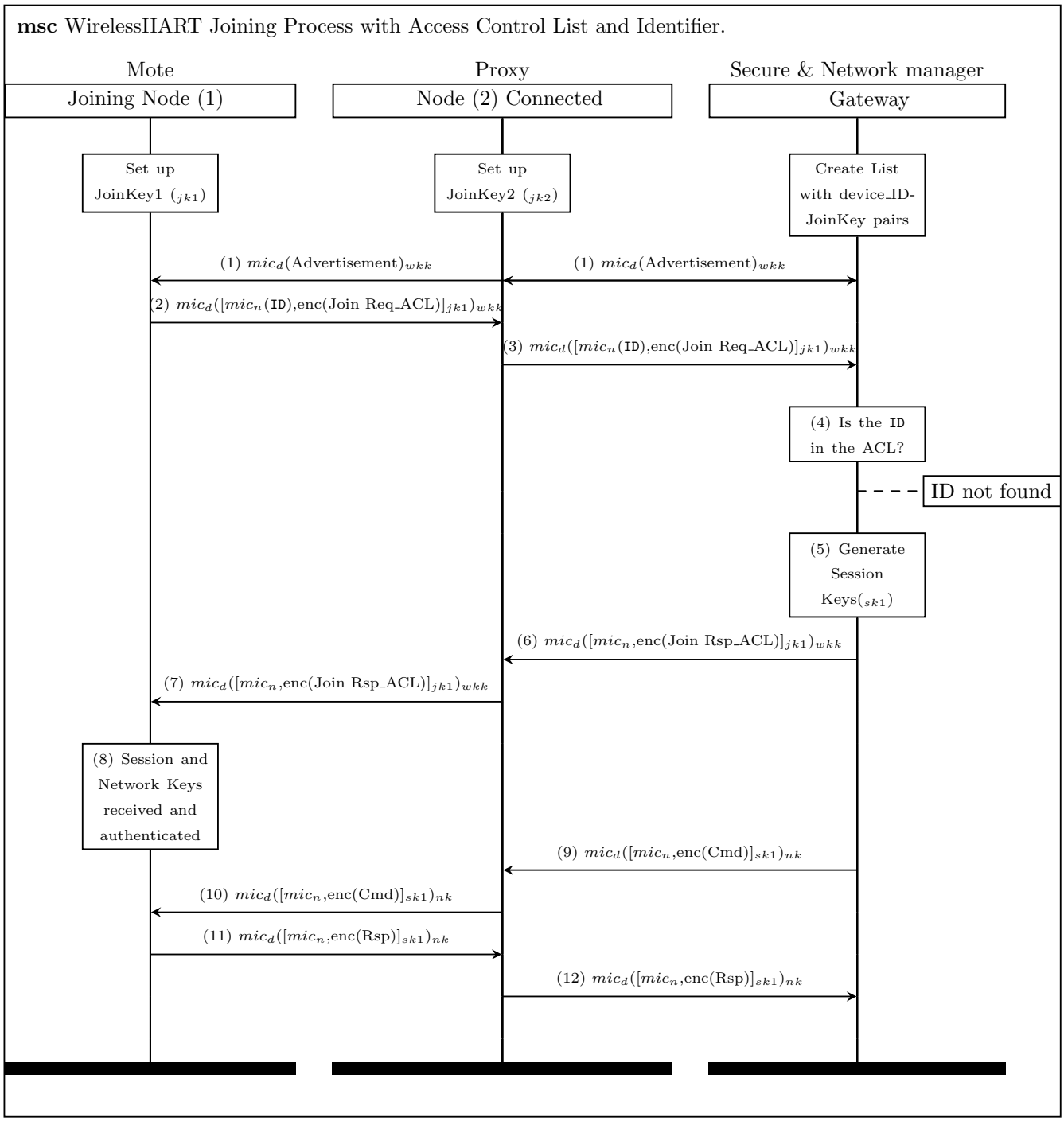
6. The GW sends the Join Response towards the unauthenticated mote through the proxy or authenticated mote.

7. The authenticated mote acts as proxy and forwards the Join Response. The intermediate is not capable of decrypting the packet because this is encrypted with a different JK which it does not own.

8. The joining node receives the session keys and the NK and it is authenticated. The security handshake is finished here.

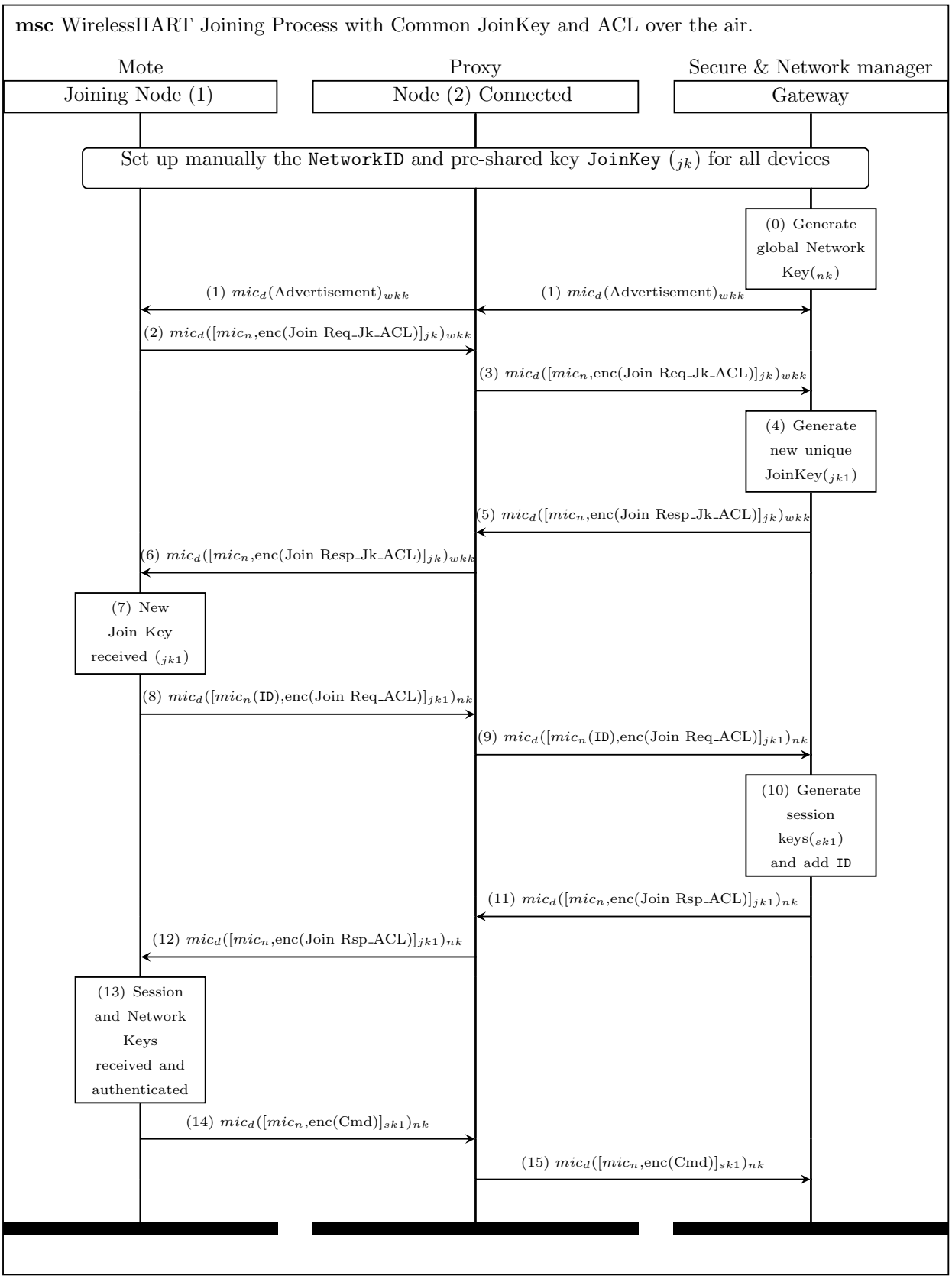
Between the steps 9-12 the WirelessHART communication starts with the normal Command-Response protocol as already explained in the previous diagram.

[16] Macaddress, serial number or any unique data could be used as IDentifier



Joining Process with Joinkey and ACL over the air

This mode is started with a common JoinKey in order to end up in an ACL mode with unique JoinKeys over the air. Once the joining process is half-finished, the unauthenticated node receives the unique Joinkey associated to it to create a final joining process over the ACL mode. The main disadvantage of this mode is the small amount of RAM of these devices for storing ACLs. Whether a node is compromised or lost, then the GW will only revoke a unique JoinKey instead all of them as used with a common JoinKey.


msc WirelessHART Joining Process with Common JoinKey and ACL over the air.
Mote
Joining Node (1)
Proxy
Node (2) Connected
Secure & Network manager
Gateway
Set up manually the NetworkID and pre-shared key JoinKey ($_{jk}$) for all devices
(0) Generate global Network Key($_{nk}$)
(1) mic_d(Advertisement)$_{wkk}$
(1) mic_d(Advertisement)$_{wkk}$
(2) mic_d([mic_n,enc(Join Req_Jk_ACL)]$_{jk}$)$_{wkk}$
(3) mic_d([mic_n,enc(Join Req_Jk_ACL)]$_{jk}$)$_{wkk}$
(4) Generate new unique JoinKey($_{jk1}$)
(5) mic_d([mic_n,enc(Join Resp_Jk_ACL)]$_{jk}$)$_{wkk}$
(6) mic_d([mic_n,enc(Join Resp_Jk_ACL)]$_{jk}$)$_{wkk}$
(7) New Join Key received ($_{jk1}$)
(8) mic_d([mic_n(ID),enc(Join Req_ACL)]$_{jk1}$)$_{nk}$
(9) mic_d([mic_n(ID),enc(Join Req_ACL)]$_{jk1}$)$_{nk}$
(10) Generate session keys($_{sk1}$) and add ID
(11) mic_d([mic_n,enc(Join Rsp_ACL)]$_{jk1}$)$_{nk}$
(12) mic_d([mic_n,enc(Join Rsp_ACL)]$_{jk1}$)$_{nk}$
(13) Session and Network Keys received and authenticated
(14) mic_d([mic_n,enc(Cmd)]$_{sk1}$)$_{nk}$
(15) mic_d([mic_n,enc(Cmd)]$_{sk1}$)$_{nk}$

The WirelessHART Joining Process with Common Join Key and ACL over the air is depicted as follows:

0. The SM creates the global NK using the JK. The SM also creates the broadcast keys using the NK.
1. The joining process starts with the advertising. All the nodes start beaconing the spectrum to offer the possibility to join the network for the unauthenticated nodes. These advertisements are signed with the WKK using AES-128-CCM*. In the diagram this appears as $\text{MIC}_d()$ and is carried out at the DLL.
2. The unauthenticated mote is awake and receives the advertisement. The mote recognises the Network ID and starts sending a Join Request. This request is enciphered with the JK and authenticated too at the network layer and authenticated with the WKK at the DLL. This Join Request specifies that the first steps will be carried out with the common JK but an ACL will be created over the air later.
3. The authenticated node acts as a proxy and forwards the Join Request to the GW verifying the MIC with the WKK.
4. The SM creates a new unique $JoinKey_1$ for the new mote.
5. The GW responds with a Join Response with the new key
6. The proxy just forwards the response to the unauthenticated mode.
7. The mote has received a new and unique Joinkey to start with a new authentication with ACL over the air as well as the global Network Key.
8. The half-authenticated mote sends a JoinRequest_with_ACL with its `ID` enciphered with the new Joinkey to the network and signed with the global Network Key.
9. Intermediary motes have to collaborate in the network. Thus this packet is relayed to the GW because this packet is proved legitimate.
10. Generate session keys (sk_1) and add the `ID` to the ACL.
11. The GW sends back the JoinRequest as JoinResponse with the session keys inside.
12. The proxy just forwards it to the destination.
13. The mote is authenticated and has all the run-time session keys to start any communication.
14. The new authenticated mote wants to send a packet to the GW by using the command-response protocol.
15. The proxy just forwards it to the destination. After that, the GW will answer the command and the communication is already started maintaining both data integrity, confidentiality and availability.

Security deficiencies in WirelessHART

Although WirelessHART was designed to be secure and reliable protocol, the current standard does have some security limitations such as a:

- The WirelessHART does not require physical device security in the standard. Despite of the fact that this protocol needs to be reliable and secure enough, there are no anti-tampering measurements to stop node capture and physical attacks.

- Since the WirelessHART protocol does not use asymmetric cryptography some security features are not achieved. For instance, non-repudiation cannot be carried out without public key cryptography.
- The key management is vaguely described and there are some gaps in the security definitions. For instance, as addressed the key management and the communication between the SM and NM is not concise.
- The entropy sources for the randomness are not specified.
- The session keys generation is not specified either, although it is known that hashing functions are involved.
- Broadcast communication between field devices is not supported.
- Once a mote is compromised, the malicious node would be able to intercept all joining requests of its neighbourhood. Owning these Join Responses, the session keys would be exposed to adversaries to intercept the end-to-end communications between network devices.

In [62], [1] and [63] these limitations are addressed as well as a prototype for a secure and scalable key renewal protocol.

2.8 Others 802.15.4 Wireless Protocols

The goal of this chapter is to address similar wireless protocols based on the IEEE 802.15.4. Although all these protocols are built upon the PHY and MAC layers, it is interesting just to address some similarities and differences between them. The next points to discuss are various specifications for higher protocol layers such as ZigBee or ISA100.

2.8.1 ZigBee

ZigBee [17] is a specification for a cost-effective, low-rate and low-power wireless communication protocol for home automation, monitoring and control defined around 2004. Besides home monitoring, Zigbee has been used for many application such as smart energy systems, telecommunication services, health care, remote control devices, retail services and other ones.

Although the ZigBee protocol has many similarities with WirelessHART, there are still many reasons why ZigBee has not been considered suitable for use in most industrial applications. An important difference to discuss in this context is mainly the security. Whereas ZigBee versions can work without security, in WirelessHART there is no way to turn off the security mechanisms. Furthermore, Zigbee was designed to offer confidentiality or integrity whereas WirelessHART offers confidentiality and integrity. On the other side, in the ZigBee protocol peer-to-peer communications can happen whereas WirelessHART such communications must reach the Gateway before arriving at the destination. In the ZigBee specification there is no frequency hopping and all the nodes share the same channel.

A comparison of WirelessHART and ZigBee for industrial applications is available in [64].

2.8.2 ISA100

ISA100 [18] is another wireless specification for industrial environments which mainly guarantees high reliability and robustness. ISA100 was developed by the International Society of Automation (ISA) around 2005. Nowadays, ISA100 is a competitor of WirelessHART claiming similar plant

[17] http://www.zigbee.org
[18] http://www.isa100wci.org

needs.

Unlike the Zigbee specification, in ISA100 both authentication and confidentiality services are independently available as in WirelessHART. The key management is basically rather similar but ISA100 provides more flexibility allowing asymmetric cryptography. In WirelessHART, asymmetric cryptography is nowadays considered either expensive in terms of power consumption or not well-implemented yet. Regarding the encryption of messages' payload and MIC calculation there is a slight difference. Such security is either applied in the transport layer (ISA100) or in the network layer (WirelessHART). Furthermore, ISA100 has an additional key, a master key that is used to generate session keys and layer-specific keys, in comparison with WirelessHART.

A notable difference between both is that ISA100 is built over IPv6 and WirelessHART over IPv4. ISA100 also supports a connection-less service based on User Datagram Protocol (UDP) packets at the transport layer. ISA100 also uses IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) at the network layer. Nevertheless, the main difference can be directly traced to the differences in the goals of each standard. WirelessHART was designed to offer reliability, security and appropriate fit for industrial environments and ISA100 was designed to provide flexibility. This flexibility provides a variety of configurations to the manufacturers for customizing the whole system [65].

A comparison of WirelessHART and ISA100 for industrial applications is available in [66, 65].

Chapter 3

Hardware Security

3.1 Overview

Previous sections have addressed basic concepts about WirelessHART and WSNs, as well as some existing approaches related to MCUs and RF modules. This section will describe the hardware security and an in-depth description of its components. First of all, the most common hardware protocols are addressed as well as the main debugging interfaces. Secondly, common hardware security protections in MCUs are discussed. Eventually, some of the physical attacks found in literature are discussed to conclude the chapter.

3.2 Hardware protocols

This section describes the main characteristics of some of the most common debugging interfaces installed on chips and boards nowadays. These debugging interfaces are designed for testing and programming chips during their manufacturing process. After manufacturing, chips need to be programmed for the resellers and this is carried out through such debugging interfaces. Once these interfaces are used to upload the firmware images, many re-sellers or manufacturers leave them opened. This represents a security issue allowing attackers to reprogram, download and debug the target devices. Protections are employed by some vendors although others are not aware that these interfaces can be a backdoor for adversaries. Along this section, a brief introduction and security issues will be addressed.

3.2.1 Serial Peripheral Interface (SPI)

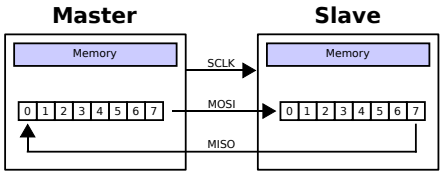


Figure 3.1: Circular buffer using two shift registers in a SPI communication.

SPI is a serial hardware protocol for embedded systems based on master-slave communication with a single master. The master always starts the communication and slaves can be selected through a slave select signal. During each SPI clock cycle, a full duplex data transmission is carried out where the master sends a bit on the MOSI line and the slave that is selected reads it, at the same time the slave sends a bit over the MISO line and the master gets the data. Slaves that have not been activated using their chip select, also known as slave select, ignore the clock cycle and MOSI lines and must not drive MISO either. The master selects only one slave at a time. The transmissions normally use a couple of shift registers, one in the master and one in the slave. This shift register is used to send bits in a circular

way between master and slave over the MOSI-MISO channel.

Many MCUs have an integrated SPI interface that is the responsible for the communication between internal components on the Printed Circuit Board (PCB). For instance, in wireless sensor nodes the communications between the main MCU and the RF chip can be carried out by using the SPI or UART protocol. Both chips must have an enabled SPI interface to communicate with.

Logic signals

SPI is also known as a *four-wire* serial bus. In order to connect through SPI, besides powering up the device, the following logic signals must be connected to the target:

Abbreviation	Description
MOSI	Master Output, Slave Input
MISO	Master Input, Slave Output
SCLK	Serial Clock
CS	Chip Select

Table 3.1: SPI logical signals.

3.2.2 JTAG

JTAG is the dominant standard for in-circuit testing for more than 20 years. This standard was designed to handle errors in digital systems such as fabrication, packaging and verification of boards. By using a JTAG programmer, it is possible to transfer data into internal non-volatile device memory. Some device programmers serve a double purpose for programming as well as debugging the device (On-Chip Debugging (OCD)). When a device is in an unknown state or 'bricked', a JTAG programmer can be used to write software and data into flash memory and recover the device. JTAG was designed to be useful for chips manufacturers and industry. However, this interface turns into a backdoor to the device when it is not well configured after manufacturing.

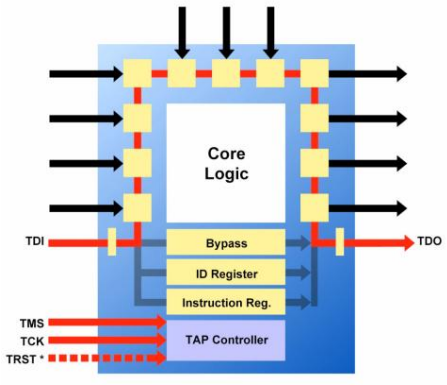


Figure 3.2: JTAG architecture. Boundary scan and TAP controller.

The JTAG protocol is defined by a bidirectional communication link following a Master-Slave protocol. A JTAG link is comprised of only one master, which always initiate the protocol and an arbitrary number of slaves. A JTAG-enabled system requires two essential physical components: a TAP controller and Boundary Scan Register (BSR). Boundary scan, shown in Figure 3.2 with black arrows, is a method for testing modern PCBs after assembly. Boundary scan can be defined as: 'the ability to set and read the values on pins without direct physical access'. A BSR is a register interposed between the chip's logic and the I/O modules. All the signals between the device's core logic and the pins are intercepted by a serial scan path known as the BSR. On the other side, a TAP is a state machine that interprets the JTAG serial protocol.

There are two kind of registers associated with boundary scan: instruction and data registers. The instruction register holds the current instruction to be carried out. Its content will be used by the TAP controller to determine what to perform with the received signals. Although there might be many registers in the standard, there are three primary data registers: [1]

- `Boundary Scan Register` (BSR). This is the main testing data register. It is used to move data to and from the I/O pins of a device.
- `BYPASS` register. This is a single-bit register that passes information from Test Data In (TDI) to Test Data Out (TDO). It allows other devices in a circuit to be tested with minimal overhead.
- `IDCODE` register. This register contains the ID code and revision number of the device. This information allows the device to be linked to its Boundary Scan Description Language (BSDL) file. The file contains details of the Boundary scan configuration for the device.

The IEEE 1149.1 standard defines a set of instructions that must be available for a device to be considered compliant. These instructions are:

- `BYPASS`. This instruction causes the TDI and TDO lines to be connected via a single-bit pass-through register (the `BYPASS` register). This instruction allows the testing of other devices in the JTAG chain without any unnecessary overhead.
- `EXTEST`. This instruction causes the TDI and TDO to be connected to the BSR. The device's pin states are sampled with the 'capture dr' JTAG state and new values are shifted into the BSR with the 'shift dr' state; these values are then applied to the pins of the device using the 'update dr' state.
- `SAMPLE/PRELOAD`. This instruction causes the TDI and TDO to be connected to the BSR. However, the device is left in its normal functional mode. During this instruction, the BSR can be accessed by a data scan operation to take a sample of the functional data entering and leaving the device. The instruction is also used to preload test data into the BSR prior to loading an `EXTEST` instruction.

Other commonly available instructions include:

- `IDCODE`. This instruction causes the TDI and TDO to be connected to the `IDCODE` register.
- `INTEST`. This instruction causes the TDI and TDO lines to be connected to the BSR. While the `EXTEST` instruction allows the user to set and read pin states, the `INTEST` instruction relates to the core-logic signals of a device.

Logic signals

The JTAG interface, collectively known as a TAP, uses the following signals to support the operation of boundary scan. The target needs to be powered up.

[1] http://www.xjtag.com/support-jtag/jtag-technical-guide.php

Abbreviation	Description
TDI	Test Data In
TDO	Test Data Out
TMS	Test Mode Select
CLK	Test Clock
GND	Ground
RST (Optional)	Reset
tRST	target Reset

Table 3.2: JTAG logical signals.

3.2.3 UART

UART is the key component of serial communications of a computer or embedded device. Basically, UART is the hardware component that converts data between parallel and serial forms. UARTs are now commonly included in many MCUs. Over these UART ports, it is possible to debug, access bootloader, program the flash memory and many other actions. From now on out, UART is also referenced as serial port in this thesis. Serial ports deal with data in byte sized pieces. UARTs asynchronously and sequentially sends out a stream of bits, one bit at a time. Conversely, the inbound transmission that enters the serial port via the external cable is converted to parallel bytes that the computer can understand. The UART protocol adds a parity bit on outbound and checks the parity of incoming bytes and discards the parity bit. Furthermore, the UART allows I/O interruptions from the keyboard and mouse in order to take over the control over the communication. It can manage other sorts of interruptions and device management that require coordinating the computer's speed of operation with device speeds.

In this thesis, it is shown that different MCUs are able to 'speak' UART between them over the serial port. Many MCUs can be sleeping until they receive a UART command. The serial port works in an asynchronous way letting wireless sensor nodes save energy when there is no activity in the network. As UART communication is present in the majority of MCUs, its hardware security is based on either the disabling of it or protecting it with a password.

Logic signals

UART is also known as serial port. In order to connect through UART, besides powering up the device, the following logic signals must be connected to the target. Since the device is already powered up, targets can get damaged when VCC is connected producing a short.

Abbreviation	Description
TX	Transmission
RX	Reception
GND	Ground
VCC (Optional)	Power-supply

Table 3.3: UART logical signals.

3.3 Hardware Security

Many manufacturers disable the UART interface before releasing devices either disabling some flags in the kernel or protecting it with passwords. Although a UART signal can be physically

active on 3.3 or $5V$, its activity is completely null if it has been previously disabled via software. Other configurations allow access to UART interfaces by entering a password. Such passwords are secret for manufacturers and can be used to debug the device in a required case. The interface JTAG is similar treated and it is either keyed with a password or disable via hardware protections.

Normally chips are programmed and subsequently disabled their debugging interfaces in order to protect their access via SPI,JTAG,... Depending on chips, there exist various measures which disable or protect the access to devices. Fuses and lock bits can be used in some architectures although other devices do not allow their programming protections. If the device's hardware does not allow to protect programming or debugging interfaces, it should not be used for any security project where sensitive information is handled.

Accessing devices should be only allowed to trusted parties. If the device is a wireless sensor node which surely will be programmed once in its lifespan, all debugging interfaces should be completely disabled via hardware. Only those projects which are in an experimental phase could protect their debugging interfaces with passwords.

This section addresses various security mechanisms used in different families of architectures. The terms 'fuses' and 'lock bits' are briefly discussed. In this thesis, our main targets have been AVR, ARM and TI MSP430. Such ICs have some built-in security measures.

3.3.1 Hardware protections: Fuses and lock bits

In general, MCUs have three memory areas: FLASH which is dedicated to program code, SRAM for run-time variables and EEPROM which can be used by user code to store data that have to be preserved when the MCU is turned off. The lock bits and fuses are the fourth non-volatile memory area available for programming. Both fuses and lock bits are preventions to avoid reading the proprietary code from chips, enable debug interfaces, disable boot settings, set the multiplier of the internal oscillator or if the reset pin can be used as a General Purpose Input Output (GPIO) pin... These lock bits and fuses should be set after programming a board or a chip itself in order to enable their hardware security.

Hardware fuses

A hardware fuse is a special area of memory that determines which and how different areas can be accessed. Applying high voltage to certain chip's pins can result in a blown fuse permanently. Hardware fuses are known for being irreversible and keeping permanently locked after their blown. For instance, whether a manufacturer provides a chip with fuses and lock bits without programming, and the chip's hardware fuses for JTAG and UART are blown, then they are locked forever.

In this research a Texas Instruments (TI) MSP430 was found without any protection. The JTAG hardware fuse was not blown and the memory content was readable by using a GoodFET programmer. Both ARM and AVR do not have hardware fuses but have software fuses and lock bits.

Software fuses

A software fuse is also a dedicated area memory where each bit is meaning the activation or deactivation of some feature. Unlike the hardware fuses, the software fuses can be programmed and reset back. Therefore, they are reversible but there is pre-condition whether someone wants to reprogram the fuses: "All data will be wiped out before setting new fuses". The erase procedure follows a certain sequence that ensures that the FLASH and EEPROM will always be erased before the lock bits are erased.

In AVR architectures, the fuses are 3 bytes containing certain information. Each bit has a different meaning depending on the chip itself. Normally fuses are programmed with logic zero and unprogrammed with high level or logic one.

- *Extended Fuse Byte.* Bit[0] and Bit[1] contain information about compatibility and watchdog timer. The rest of bits are seldom used. But each chip has a different configuration.
- *Low Fuse Byte.* The low fuse byte has different options such as: clock sources, brownout detector trigger level and start-up times (`SUTx,CKSELx, BODEN`).
- *High Fuse Byte.* This byte is using all the bits and it is the most interesting fuse byte to find out if there is some possibility of entry in the system. These fuses might have different bit's position and thus are chip-dependant. However, many chips share the most important in different order. The meaning of activation of such fuses is discussed below.

  - `OCDEN`. Enable the On-Chip Debugging OCD. This feature is expensive in terms of power consumption and it is disabled normally. Default value: Disabled (1).
  - `JTAGEN`. Enable the JTAG interface. Default value: Enabled (0).
  - `SPIEN`. Enable the SPI interface. Default value: Enabled (0).
  - `EESAVE`. EEPROM memory is preserved after chip erase. Default value: Disabled (1).
  - `DWEN`. Enable the *DebugWire* interface. Default value: Enabled (0).
  - `BOOTSx`. Select boot block size. Default value: Enabled (0).
  - `BOOTRST`. Select the reset vector. It makes program execution start from the bootloader section rather a from normal flash start (0x0000). Default value: Disabled (1).
  - `CKOPT`. Oscillator options and speed of the clock. Default value: Disabled (1).

When `JTAGEN, SPIEN, DWEN` are wrongly set then the only chance to recover a chip after occasionally setting these bits is to restore the default settings with a parallel programmer High Voltage Parallel Programming (HVPP).

Lock bits

The lock bits are a similar concept as fuses. Lock bits are mainly used to protect the memory access and boot loader. A special memory area with some bits determining certain program memory features. Depending on the chip type, there can be different numbers of lock bits. But the two least bits are always present. `LB1` and `LB2` bits are used to lock the memory content. The boot lock bits `BLB01, BLB02, BLB11` and `BLB11` are used to lock writing and reading to/from FLASH memory either from application area or bootloader section.

When `LB1` is programmed, further programming of Flash and EEPROM is disabled in high-voltage and serial programming modes. Fuse bits are locked in both serial and high-voltage programming modes. When both `LB2` and `LB1` are programmed, further reading and programming of Flash and EEPROM is disabled in high-voltage and serial programming modes. Fuse bits are locked in both serial and high-voltage programming modes.

3.3.2 Bootstrap Loader (BSL)

The BootStrap Loader (BSL) is an application built into TI MSP low-power MCUs. It enables the user to communicate with the device to read from and write to its memory. This feature is primarily used for programming the device, during prototyping, final production, and in service. Both the FLASH and RAM can be modified as required through the BSL. The BSL is password-protected with a password of 32-byte characters long. The BSL password is the value of the Interrupt Vector Table (IVT) of the chip, which resides at the top of memory and is composed of

pointers to interrupt handlers. Concretely the chip TI MSP430 has two hardware security features: JTAG and the BSL. Whether the hardware fuse JTAG is blown, then the only access to the chip is through the BSL.

Goodspeed [67, 68] presents research in a Side Channel Attack (SCA) attack in the MSP430 BSL. Goodspeed provides a board called BSLCracker which is able to reduce the bruteforce attack. However, the most recent attack is carried out with the flash of a current camera. The "paparazzi" attack is targeted a physical key container called SupraBox [69]. The reason of this attack is due to the ultraviolet rays shot with the flash of the camera. Chips are susceptible of these kind of attacks and ultraviolet light can defeat this security mechanism.

In this thesis it was not necessary to attack the BSL since the JTAG interface for this MCU was opened. If it was opened, deccaping techniques would have been needed to successfully extract the memory content.

3.4 Physical Attacks on Wireless Sensor Networks

In 2005 [70], American researchers proposed several hypothetical attacks through node compromise attacks. They highlighted the importance of such attacks for instance in military installations and scientific labs. Authors of this research showed how they were capable of extracting the memory content by using the debugging interfaces (JTAG, UART). In the paper, authors ran an experiment with the well-known *Mica2* and the *AVR ICE* JTAG. Not only the memory or *EEPROM* content was dumped, moreover they were able to extract the SRAM memory in a matter of seconds. Several experiments led to obtain the cryptographic keys in all the attempts. Either because the keys were always stored at the same location, or because keys were stored in plaintext in the flash memory. Finally authors claim that the longest task only took around 30 seconds and concluded at the time of writing that WSNs designers cannot trust only the software security measurements or the secure protocols of symmetric cryptography. Although sensors use rather secure AES encryption, the hardware part always needs to be protected carefully. To prevent this problem the authors state that the only solution is tamper proof hardware which triggers some type of self destruct mechanism upon attempted compromise.

In 2008 [71] and [72], Travis Goodspeed published an academic paper on how to reduce a bruteforce attack against the BSL on MSP430 TI chips. A 32-byte password is generated from the IVT and it is the only way to access to the chip when the fuses were blown. This side channel timing attack is shown with a home-made physical board, BSLCracker, which is able to show the timing leakage during the authentication and perform the attack. In 2009 [73], the same authors showed how to exploit low-powered wireless embedded systems. GoodSpeed provides a reverse engineering research focussed both in hardware and software. The author is capable of sniffing buses in wireless nodes and recovering the AES cryptographic keys in matter of seconds for the well-known CC2420 radio chips. Furthermore, an explanation of how to access firmware images in TI MCUs is carried out. Some memory mapping techniques are shown be useful to identify possible functions on the proprietary code as well as more software attack vectors such as stack overflows, fuzzing techniques or how an infected node could act as a worm in the network. Francillon and Goodspeed also run experiments on how to half-blind recover the firmware image of a MCU. In this paper, authors emphasise the fact of the existence of a bootloader ROM to break the security of a MCU by a Return Oriented Programming (ROP) attack. Main advantages of these attacks rely on the use of 'gadgets' or existing code to execute code on the target.

In 2013 [74] a paper was released explaining some vulnerabilities affecting industrial wireless automation protocols such as ZigBee or WirelessHART. First of all, authors reverse-engineered a binary responsible for generating the default JoinKeys for an unknown-vendor. The problem was due to the low entropy level of the Pseudo Random Number Generator (PRNG) since developers

used the Linux Epoch time as seed for the random functions leaving a possible way for bruteforce attacks. Generating the whole range of AES-128 keys between 1970/01/01 and today would allow any adversary to recover the Joinkey in matter of minutes. Another flaw discovered in another devices relied on the 'Project File', a file responsible for managing the security keys. Changing the file, a new key was generated and subsequently distributed over the sensor nodes. Authors applied *binary diffing* techniques to figure out which offsets were altering the security keys. Same issues are reflected in several examples more along this paper. The final conclusions mainly suggest to enforce physical security using anti-tampering mechanisms, audit the source code, never trust vendor's documentation and use secure out of band methods for distributing the keys.

Giacomo et al. [75] also comment on possible consequences when a node capture occurs by analyzing the power consumption in a couple of wireless sensor nodes. Authors work on SCA attacks in two cryptographic implementations: AES and Elliptic Curve Cryptography (ECC). Although there are complex settings in the laboratory, authors suggest to protect sensor nodes with low-cost side-channel countermeasures to avoid any kind of leakage. In 2008 [76], 2013 [77] some techniques were addressed in order to strengthen the security of the WSNs. This paper is focussed on the resistance of wireless nodes against non-invasive attacks known as SCA.

3.4.1 Consequences and possible attacks

In this thesis we address the following questions:

1. Can an adversary extract the JoinKey from the mote without interrupting WirelessHART communications, without being seen nor detected and subsequently he/she being able to be authenticated in the WirelessHART network?.

2. What would happen when an adversary is capable of physically stealing a node in the network and bring it to his/her laboratory to be dissected?

3. When the motes have security protections, are they enabled by default?

The first question has several points of view and it depends on the configuration in the wireless network. There are WSN, that have smart and efficient algorithms to detect internal problems. An important internal issue is the '*sinkhole*' attack that occurs when a compromised node attracts network traffic by advertising fake routing updates. Basically what it happens is that the node claims to know the shortest path to the base station and attracts as much traffic as possible to itself. Obviously this can provoke denial of service in a big portion of the wireless network for a long time. The WSNs are known to be vulnerable to this attack due to the ad-hoc network structure and many-to-one connections between nodes and base station [78]. Targeting the closest node to the base station would produce a huge impact in the performance of the wireless sensor network. One of the most severe attacks to detect and defend in wireless sensor network is a '*wormhole*' attack. Such attack occurs when a node is compromised and creates a tunnel with another malicious node and the packets are tunneled between them in such a way that tunneled packet arrives sooner than the normal packets relayed over a hop-to-hop route and eventually the fake route is selected first. The wormhole puts the adversary in a very powerful position relative to other nodes in the network and the adversary could exploit this position. This attacks is considered rather hard to stop in real life. Detecting sinkhole [79, 80] and wormhole [81, 82] attacks can result costly and extra measures must be set up.

The second question is considered as the node leaves the environment and an alarm is rapidly triggered in the network. Depending on the security configuration, this node capture attack could be detected and stopped on time. Both sinkhole and wormhole attacks are available in this scenario as well when there exist deficiencies in the security level. This scenario is considered less impressive due to the security alerts.

The third question can be answered without going further. Since our first target, the Linear Kit, is a development kit the security protection is disabled by default to be as more user-friendly as possible. As it has been proved in this paper, the hardware protection was disabled by default in the Linear Kit and thereby cryptographic keys were extracted in matter of seconds.

As explained in the Chapter `Background`, communications in both PHY and DLL are enciphered with AES-128-CCM* and they cannot be understood even whether an adversary is capable of eavesdropping the communication. The only reasonable way to understand this data is owning the JoinKey to be authenticated into the network and intercept join responses bringing run-time session keys. This thesis describes several ways to obtain this cryptographic AES Joinkey.

Chapter 4

Tools

4.1 Outline

In this section tools used both in hardware and software are briefly outlined and described. This section aims to give an impression of the adversaries' budget and an idea of which tools have been employed. The author of this thesis considers the budget for this set of tools affordable for a motivated adversary. Although more sophisticated tools could facilitate the task, this study is set out to use tools as cheap and accessible as possible for any adversary. Expensive equipment is out of the scope and the author has chosen the most well-known open-source hardware and software tools. Exceptions with the disassembler and decompiler. For instance, the author considers soldering iron, multimeter, wrapping cables, magnifier, helping-hands and basic stuff included in our laboratory. Nevertheless, a hot air rework station was not available in this set up and it would have been rather useful. This section sketches out a couple of JTAG programmers for the most important MCUs' architectures for WSNs: AVR and TI.

4.2 Bus Pirate v.3.6

The Bus Pirate is a universal electronic open hardware tool capable of performing a number of hardware protocols. It is informally defined as an 'open source hacker multi-tool that talks to electronic stuff' [1] by its designers. This hardware tool has been used to talk to the majority of protocols found in the MCUs involved in this thesis. Principally, the most used protocol has been SPI. The Bus Pirate supports protocols such as: UART, SPI, I2C, JTAG, 1Wire, MIDI, PC keyboard and many others. Although other programmers are working much faster for readings and writings, the Bus Pirate accomplished a lot of our necessities for the amount of €25.

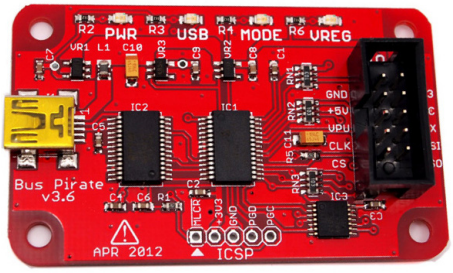


Figure 4.1: The open source hardware hacker tool 'Bus Pirate' v3.6 used for interacting with chips.

On the software side, the Bus Pirate offers an open source framework which is able to speak to several protocols. The firmware also depends on the version of hardware used and it can allow us to carry out more or less actions depending on its version. For instance, only some versions permit using JTAG functionalities. Although the Bus

[1] http://dangerousprototypes.com/docs/Bus_Pirate

Pirate already provides a framework, external applications also support this hardware. In this thesis, the software Flashrom was used in combination with the Bus Pirate in order to speak SPI to chips. On the hardware side as showed in Figure 4.1, the Bus Pirate is comprised of a PIC24 MCU and a USB interface with a FT232RL as well as a couple of pin-headers for programming and debugging purposes. Our normal set up is as follows: a PC connected to the Bus Pirate over the USB protocol, and the Bus Pirate attached to the target either with pins or soldering some cables to the target's pins.

4.3 GoodFET v.42

The GoodFET [2] is an open source JTAG programmer based on the TI MSP430 Flash Emulation Tool (FET) debugger [3] and EZ430U boards. This JTAG programmer is able to communicate with many wireless sensor nodes which use a TI MSP430 chip. On top of these chips, GoodFET is also able to operate with some Chipcon radios with embedded 8051 cores such as the CC2430 and CC2530. It is also able to work with Nordic and Zensys (Z-wave protocol) radios as well as to handle the SPI protocol for numerous memories and chips. As it is open source, the community is constantly developing new software for new chips and the number of supported chips may grow.

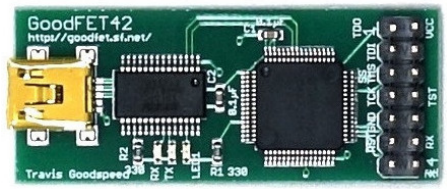


Figure 4.2: The open source JTAG programmer 'GoodFET' used for debugging several wireless radio chips.

On the software side, the GoodFET has an internal firmware written in C, which is responsible of managing the communication with chips and also communicates with the client. The GoodFET client is written in Python and provides different commands for the aforementioned chips. On the hardware side as showed in Figure 4.2, the GoodFET has an MSP430F1618 as main MCU and a TUSB3410 usb to serial converter. Basically, the GoodFET is connected over USB to our computer and attached to the target through its JTAG ribbon cable.

4.4 TL866A USB Universal Minipro Programmer

Since memory dumping is one of the most basic attacks considered, an EEPROM programmer will be necessary. As explained in the previous chapter, some wireless sensor nodes are comprised of storage units to maintain the environment data. In the normal case, there is either an external memory to store measurements or the on-board flash memory in the MCU to store this data. An external memory is normally used because the lifespan of flash memories is limited. A constant writing into the on-board flash memory would reduce the lifetime of the wireless sensor nodes.

Our budget would not allow us to buy an expensive EEPROM programmer worth more than €200 without adapters. Nevertheless, a really good price was found in a Chinese EEPROM reader worth around €70 which supports more than 13000 chips. Besides, in this price many of the most common adapters were included in the shipment. As the newer version TL866CS did not support the In Circuit Serial Programming (ICSP) interface to program SPI memories, the previous TL866A version was our choice. In many occasions this EEPROM programmer verified the SPI readings carried out with the Bus Pirate or vice versa.

[2] http://goodfet.sourceforge.net/
[3] http://www.ti.com/lit/ug/slau278u/slau278u.pdf

4.5 AVR Dragon JTAG programmer

Since there are many wireless nodes comprised of AVR architecture's MCUs, a programming interface is also required. In our set up the AVR Dragon has been picked mainly due to two features: the price and the amount of possible programming's manners. This hardware tool also offers support for the following programming interfaces: SPI, High Voltage Serial Programming (HVSP), Parallel Programming (PP), JTAG, Program and Debug Interface (PDI) and *aWire*. For debugging purposes the AVR Dragon supports JTAG, *debugWire*, PDI and *aWire*. Some of these interfaces are Atmel proprietary and have not been used in this research. Besides ICSP and JTAG programming interfaces, the AVR Dragon offers a header for HVPP. This interface is the only solution when both SPI and JTAG have been enabled and it is not possible the access the MCU. This feature is dependant on the MCU although many of them have this feature. The AVR Dragon supports all programming modes for the Atmel AVR device families. The AVR Dragon can be acquired for approximately €50.

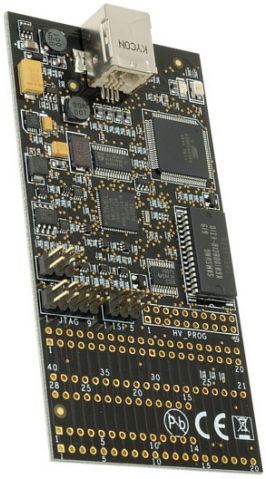


Figure 4.3: The Atmel AVR Dragon programmer.

On the software side, the AVR Dragon can be used with different software tools including the official Atmel Studio and some open source tools such as avrdude [4] or AVaRICE [5]. On the hardware side the AVR Dragon is mainly composed of two Atmel MCUs: ATMEGA2560 and ATMEGA128A. The AVR Dragon is connected to our PC over the USB protocol.

4.6 ARM Segger J-Link JTAG programmer

Nowadays, the most common architecture for embedded systems is principally the ARM-based platform. The majority of wireless sensor nodes and Managers are running on ARM architectures. Our lab has an ARM J-Link JTAG programmer. This JTAG programmer is manufactured by Segger and an EDU version can be used for non-commercial purposes, having the same functionalities as the J-Link BASE model. The MCUs supported for this JTAG programmer are: any ARM7/9/11, Cortex-A5/A8/A9, Cortex-M0/M1/M3/M4/M7, Cortex-R4, Microchip PIC32 and some Renesas ones. The J-Link EDU version has a final price of around €60.

On the software side, this JTAG programmer has its own software from Segger. However, it can be used with many other open source tools such as OpenOCD, Atmel Studio, Keil or Crossworks. The J-Link JTAG programmer has a JTAG and USB port in it. This programmer is connected over USB and attached to the target through a JTAG ribbon cable.

4.7 Open On-Chip Debugger

The OpenOCD allows to provide debugging, in-system programming and boundary-scan testing for embedded devices. To run this software, OpenOCD needs a debug adapter to interact with the

[4] http://www.nongnu.org/avrdude/
[5] http://avarice.sourceforge.net/

target - called dongles. There are plenty of debug adapters which are supported by OpenOCD: USB-based, parallel port-based and other standalone boxes that run OpenOCD internally. A debug adapter is any device with a feature like On-Chip Debugging, programming interface or boundary-scan. With this tool a user is capable of halting the MCU in order to debug it through the GDB protocol, programming a new memory content into the flash memory, adding hardware breakpoints at certain points or checking boundary-scan in devices.

4.8 Flashrom

Flashrom [6] is an open source utility able to interact with an enormous amount of flash chips. The main operations depend on the chip itself and its development's state. Between such operations we find reading, writing, verifying, erasing and identifying flash chips. At this time, Flashrom "supports more than 450 flash chips, 286 chipsets, 450 mainboards, 75 PCI devices, 13 USB devices and various parallel/serial port-based programmers" and thus is increasing every day. Regarding the sockets, Flashrom supports parallel, LPC, FWH and SPI flash interfaces and various chip packages (DIP32, PLCC32, DIP8, SO8/SOIC8, TSOP32, TSOP40, TSOP48, Ball Grid Array (BGA) and much more. Flashrom supports many different programmers, including PC mainboards, various PCI cards with soldered-on flash chips, and various USB/serial-port/parallel-port based programmers [7].

4.9 IDA Pro and Hex-Rays

The Interactive Disassembler, more commonly known as IDA, is a disassembler for computer software which generates assembly language source code from machine-executable code. Many of the main CPUs used for wireless sensor nodes are well supported, amongst them ARM, MIPS or TI MSP430. IDA performs automatic code analysis for any of the aforementioned architectures. Hex-rays is a very handy plugin for IDA, it allows to decompile ARM machine-executable code into C-like source code. This tool has been mainly used to generate C-like code from ARM binaries dumped from some WirelessHART devices. Although there are open source alternatives such as *Radare* [8], the Hex-rays plugin converts the tool in a fantastic way to obtain 'source code' from memory dumps in ARM devices. The IDA Pro license plus Hex-rays plugin is rather expensive; the author was fortunate to be using a valid company licence. If this option was not possible, *Radare* could also be used without the possession of an IDA Pro licence.

[6] http://flashrom.org
[7] http://flashrom.org/Supported_programmers
[8] http://www.radare.org

Chapter 5

Targets

5.1 Outline

This section addresses the two WirelessHART nodes which have been analyzed in this thesis. First of all, the Linear SmartMesh WirelessHART Starter Kit is presented. This target is a development board to experiment with the WirelessHART protocol. This starter kit has been rather helpful in order to find out how WirelessHART works. The Linear kit provides APIs, source code, datasheets and much more information about the WirelessHART environment. Secondly, an `unknown-mote` is presented and analyzed. Some details about the `unknown-mote` will be shown in this thesis, but some other specific details will remain out of the public version of this thesis.

After presenting the two targets, several attempts to extract both the cryptographic keys and firmware images are technically explained. As saw in previous chapters, the main goal is to dump the content of the MCUs' memories where the cryptographic keys normally reside. To do so, various techniques will be addressed and discussed. The readers should keep in mind that neither the BGA socket for a flash programmer nor a rework station were available during this research. Since our best guess is that the *JoinKey* is stored in the internal and non-volatile memory from the RF module. Skipping this check, the likelihood of success decreases a lot.

This chapter concludes with the goal achieved for the Linear Kit but not for the `unknown-mote`. In the first one it is clearly demonstrated how to retrieve the cryptographic keys out of the device due to its default configuration. In the latter, the cryptographic keys are not extracted, but there are possibilities of achieving this goal with a larger setup and funding for hardware tools. Although the Linear Kit seems less secure than the `unknown-mote`, it is actually the other way around. The `unknown-mote` has no hardware fuses in its RF chip whereas the Linear Kit is an improved version of the `unknown-mote`'s chip. This fact can reflect the following question: 'Why the keys were not extracted if the `unknown-mote` had less security?'. This question is again answered through this chapter. Both the BGA package and non-experience with such package paralyzed many attempts of reading the keys out.

5.2 Linear SmartMesh WirelessHART Starter Kit DC9007

A DC9007 SmartMesh WirelessHART Starter Kit [83] was given in order to get started with WirelessHART. This kit is shown in Figure 5.1 and includes the following components:

- 5 development motes (DC9003A-C) powered with button batteries (CR2032).
- 1 AP Manager (LTP5903CEN-WHR).
- 1 programming interface called Eterna Interface card (DC9006).

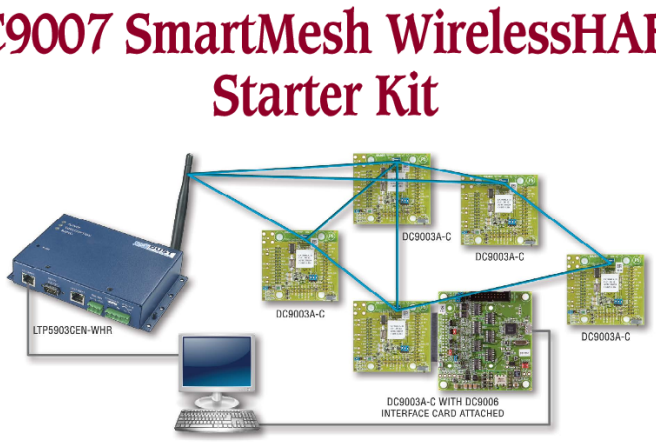


Figure 5.1: Dust Networks/Linear SmartMesh WirelessHART Starter Kit

Our first target is clearly the Linear mote although the reverse-engineering research in the AP manager might provide us with some clues on how the WirelessHART protocol is implemented in the Linear devices. The given programming interface allows us to communicate with the mote through the Linear API.

5.2.1 Hardware Components: Mote DC9003A-C

The Dust/Linear mote is a single MCU with on-board components such as RAM, RF and flash. The mote holds the following features:

- `CPU`: Eterna LTC5800WR. Mote-on-chip architecture ARM-Cortex-M3 32 bits
- `Non-volatile Memory`: Flash On-chip 512 KB
- `Memory`: On-chip SRAM 72 KB
- `Clocks`:
  - 32kHZ crystal (a low power oscillator)
  - 20MHz crystal (radio reference oscillator)
  - an internal relaxation oscillator
- `Hardware Crypto-Engine`: AES-128 bits
- `Hardware Lock Key`: Fuses table [1]
- `Radio 802.15.4`: Radio modulation and demodulation module.

As Figure 5.2 depicts all components integrated in a single chip. This chip holds all components above mentioned and others as well as analog-digital or digital-analog converters and voltage regulators. As all data is encrypted and decrypted, the Linear chip has an internal hardware cryptographic engine in order to execute fast AES operations via hardware. Doing so, it is possible to save enough memory to normally keep handling software processes. Furthermore, we can also distinguish possible debugging interfaces to connect with such as SPI and UART. Further information about the hardware design [2].

[1] http://cds.linear.com/docs/en/design-note/Board_Specific_Configuration_Guide.pdf
[2] http://www.linear.com/docs/42503

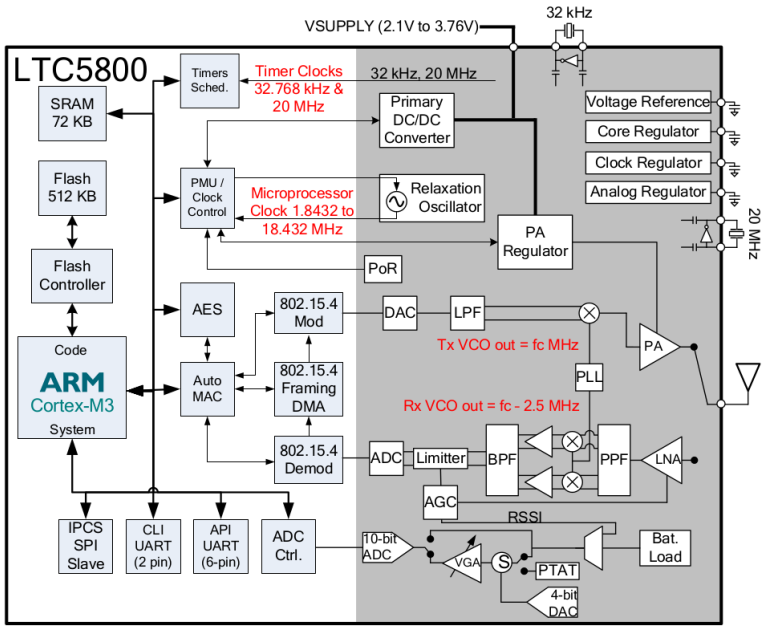


Figure 5.2: Linear Mote DC9003A-C architecture and its ARM Cortex M3 Eterna LTC5800.

5.2.2 Setting up a WirelessHART network

Once this kit was obtained, a WirelessHART network was created in our laboratory to play around with the network and use the tools offered by Linear [3]. To create this network, all the motes need to be set up with the same *NetworkID* and *JoinKey*. All devices can be accessed through the serial port or `SSH` and several APIs are helping the user to modify this data. The AP Manager also provides a web server and can be reached through a password-protected web interface. Once all devices are configured with the same *NetworkID* and *JoinKey*, the joining process is carried out and eventually the WirelessHART network is considered functional. At this moment, all motes can be set up with a known AES *JoinKey* and *NetworkID*. With this setting, our goal is now to find out where this data is stored.

5.2.3 Identifying programming interfaces in the Mote DC9003A-C

First of all, in order to identify possible accesses to the system, the reading of datasheets is the first step. After reading datasheets from the manufacturer, various programming interfaces are detected and verified. Both JTAG, SPI and UART are enabled by default. This specific System-on-Chip (SoC), also called Mote-on-Chip, has only built-in components. Therefore, our target is the SoC Eterna, which holds on-board flash, RAM and WirelessHART module. Connections to the Linear mote can be carried out using a USB programming board as Figure 5.3 shows or connecting the wanted device to the mote through its pins.

[3] http://www.linear.com/products/smartmesh_wirelesshart

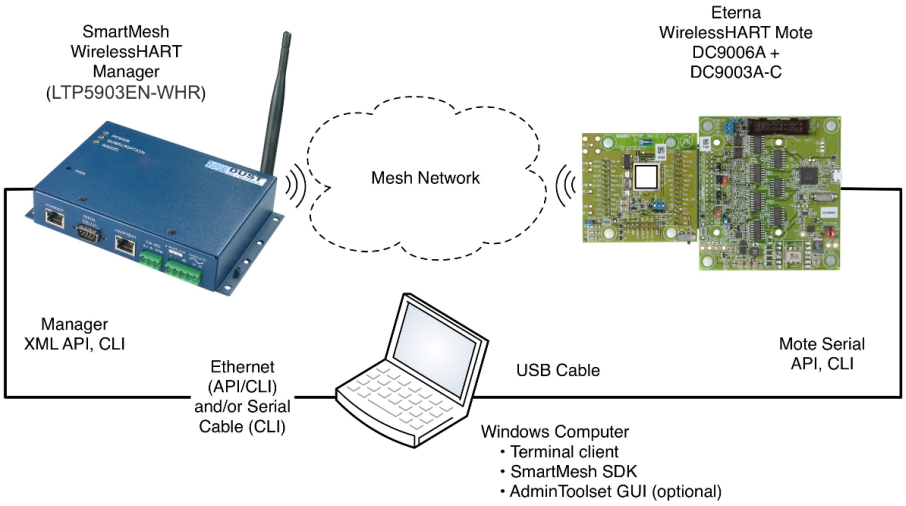


Figure 5.3: Linear WirelessHART Kit set up.

5.2.4 UART

The UART interface is used to connect over serial port; a constrained command line interface allows us to check the current settings. It is possible to modify the *NetworkID* and *JoinKey*, but the *JoinKey* is not displayed. Basically the shell is not providing access to the reading of the *JoinKey*. However, it allows us to set up many other parameters in a handy way. Although the UART allows access to the device, apparently the *JoinKey* is not accessible from this interface.

```
> mset netid 1337

> mget netid
netid = 1337

> set mode slave

> mset jkey deadbeefdeadbeefdeadbeefdeadbeef

> minfo
HART stack ver 1.0.2 #1
state:       Search
mac:         00:17:0d:00:00:60:13:c7
moteid:      0
netid:       1337
blSwVer:     15
ldrSwVer:    1.0.3.12
UTC time:    6842:6842
reset st:    0x100

> mget jkey
Parameter jkey is hidden
```

Figure 5.4: Setting up a JoinKey through the Linear Mote Eterna API over the UART interface.

5.2.5 JTAG

The first attempt of recovering the AES JoinKey was carried out via JTAG. Since our target was not supported by the software tool for JTAG,OpenOCD, some modifications were carried out in order to declare the FLASH, a TAP interface and the architecture. Instead of using the Linear tools, OpenOCD and the J-Link Segger JTAG were used. After struggling with OpenOCD's configuration, it was possible to communicate with the hardware through JTAG protocol. Once halted the CPU and previously declared a flash bank, a TCP socket is responsible to interact with the hardware. All commands sent to the hardware with the JTAG enabled and SoC halted interact with the real hardware. Our first attempt is to dump the whole memory content performing certain commands. Unfortunately, several errors were not possible to fix and as there were other debugging interfaces to interact with; JTAG debugging was temporary abandoned. Figures 5.5 and 5.6 show further details.

```
$ openocd -f openocd.cfg
Open On-Chip Debugger 0.8.0 (2014-10-13-10:50)
Licensed under GNU GPL v2
Info : only one transport option; autoselect 'jtag'
adapter speed: 10 kHz
adapter_nsrst_delay: 200
jtag_ntrst_delay: 200
cortex_m reset_config sysresetreq
Warn : Specify TAP 'ltc5800wr.cpu' by name, not number 0
trst_and_srst separate srst_gates_jtag trst_push_pull srst_open_drain connect_deassert_srst
Info : J-Link initialization started / target CPU reset initiated
Info : J-Link ARM V8 compiled Nov 25 2013 19:20:08
Info : J-Link caps 0xb9ff7bbf
Info : J-Link hw version 80000
Info : J-Link hw type J-Link
Info : J-Link max mem block 9296
Info : J-Link configuration
Info : USB-Address: 0x0
Info : Kickstart power on JTAG-pin 19: 0xffffffff
Info : Vref = 3.649 TCK = 1 TDI = 0 TDO = 1 TMS = 0 SRST = 0 TRST = 0
Info : J-Link JTAG Interface ready
Info : clock speed 10 kHz
Info : JTAG tap: ltc5800wr.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x4)

   TapName             Enabled  IdCode     Expected   IrLen IrCap IrMask
-- ------------------- -------- ---------- ---------- ----- ----- ------
 0 ltc5800wr.cpu          Y     0x4ba00477 0x4ba00477     4 0x01  0x03
Info : accepting 'telnet' connection from 4444
```

Figure 5.5: JTAG'ing the Eterna Mote-on-chip with OpenOCD and J-link ARM JTAG USB.

```
edu@foxit:~$ telnet localhost 4444
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Open On-Chip Debugger
> halt
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x81000000 pc: 0x000c9b96 psp: 0x200010c4
> flash banks
#0 : ltc5800wr.flash (lpc2000) at 0x00000000, size 0x00080000, buswidth 0, chipwidth 0
> flash info 0
#0 : lpc2000 at 0x00000000, size 0x00080000, buswidth 0, chipwidth 0
        #  0: 0x00000000 (0x1000 4kB) protected
        #  1: 0x00001000 (0x1000 4kB) protected
    <SNIPPED>
        # 28: 0x00070000 (0x8000 32kB) protected
        # 29: 0x00078000 (0x8000 32kB) protected
lpc2000 flash driver variant: 2, clk: 4000kHz
> flash protect 0 0 29 off
cleared protection for sectors 0 through 29 on flash bank 0
> dump_image moteFlash512kb.bin 0x0 0x00080000
JTAG-DP STICKY ERROR
MEM_AP_CSW 0x23000062, MEM_AP_TAR 0x1004
Failed to read memory at 0x00001004
> dump_image maxSize.bin 0x0 0x00001000
dumped 4096 bytes in 14.105293s (0.284 KiB/s)
```

Figure 5.6: Connecting to the TCP socket and attempting to dump the flash on the Eterna Mote-on-chip.

5.2.6 SPI and the Hardware Lock Key

Since the JTAG access was apparently ending up in continuous errors, a new attempt was carried out over the SPI interface. In order to successfully read the flash content over SPI, the hardware lock key has to be unlocked. This hardware lock key has the same function as a fuse. This hardware lock key [4] is constructed with 4 bytes of hexadecimal data. It is considered unfeasible to run online bruteforce attacks due to their slowness. It is well-known that online attacks against the mote might be costly on time. If this hardware lock key is enabled, the flash memory will not be readable until the flash memory is unlocked.

5.2.7 Firmware and JoinKey extraction

First of all, several ways are shown to extract both the cryptographic AES Joinkey and the whole firmware image from the on-board flash.

Linear Eterna Mote

The more straightforward method to extract the flash content was over the SPI interface. The SPI reading was carried out by using different software and hardware. The following points describe the settings used:

- `Eterna Serial Programmer` and the Linear tool "`ESP.exe`". An easy attempt was to use the given programming interface called Eterna Interface card (DC9006) with the official software provided by Linear. The connection used is depicted in Figure 5.3 which shows the USB programming interface attached to the mote and also connected to the PC over USB. Within seconds the memory content was dumped into a binary file. Figure 5.7 shows both the command and the memory offset when the *JoinKey* was found. For this configuration, neither soldering nor software skills are needed. The Eterna serial programmer is easily connected to the mote and the binary provided is statically compiled with all the libraries included.

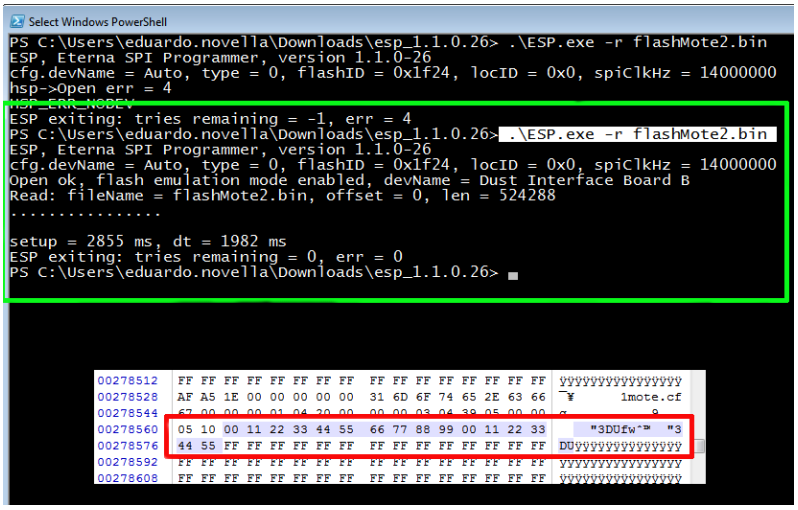


Figure 5.7: Dumping the mote's memory content through SPI using Linear tools.

- `Bus Pirate v3.6` and `flashrom`. After achieving it with the official tools, we tried to perform the same attack but with open source tools. To do so, flashrom in combination with the famous Bus Pirate v3.6 reproduce the same result. This configuration is depicted in Figure

[4] http://cds.linear.com/docs/en/software-and-simulation/040-0110rev9_Eterna_Serial_Programmer_Guide.pdf

5.8. A downside is the reading speed, whereas the official tools were finishing in around 3 seconds, the Bus Pirate uses almost 1 minute to read the whole flash memory. The Bus Pirate is known to be slow and surely another SPI programmer would read the flash memory in matter of seconds. The precise command is depicted in Figure 5.9. After run flashrom, we figure out the manufacturer of the flash memory. Apparently it seems to be an Atmel AT45DB041D [5].

Finally, Figure 5.10 shows a highlighted piece of flash memory containing all the previous *JoinKeys* used in the mote as well as the actual one. The *JoinKeys* are detected by identifying the sequence of 2 bytes: "`05 10`" and the following 16 bytes are AES keys of 128 bits.

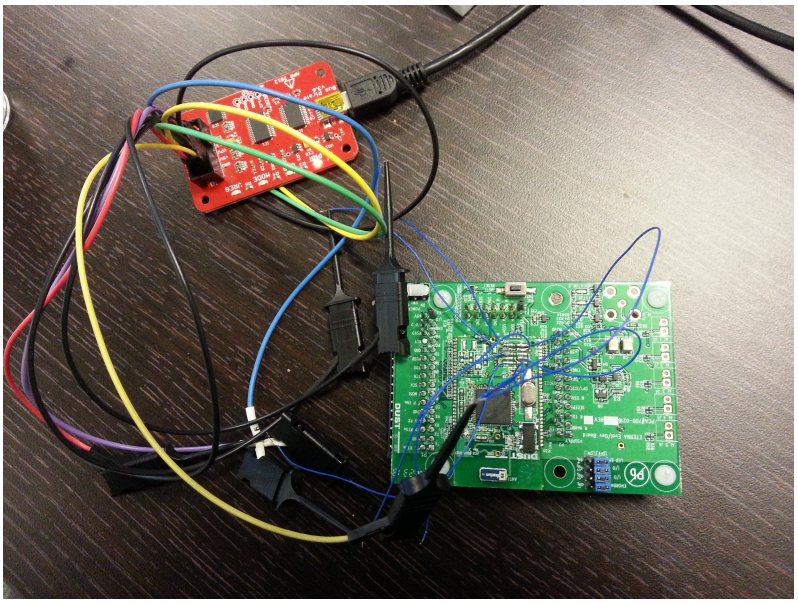


Figure 5.8: Dumping the mote's memory content through SPI using a Bus Pirate v3.6

```
$ time sudo flashrom -p buspirate_spi:dev=/dev/ttyUSB0,spispeed=1M -r MoteEterna512kB.bin
flashrom v0.9.7-r1869 on Linux 3.13.0-44-generic (x86_64)
flashrom is free software, get the source code at http://www.flashrom.org

Calibrating delay loop... OK.
Found Atmel flash chip "AT45DB041D" (512 kB, SPI) on buspirate_spi.
Reading flash... done.

real        0m54.804s
user        0m6.199s
sys         0m48.022s
```

Figure 5.9: Linear Mote Eterna firmware extraction by using Flashrom and Bus Pirate v3.6

- `TL866A EEPROM programmer`. Performing the same reading with an EEPROM programmer concludes with the extraction of the same memory content in a matter of seconds as well. Concretely we used the TL866A programmer through ICSP pins.

[5] http://www.atmel.com/images/doc3443.pdf

```
$ hd MoteEterna512kB.bin |egrep -i "05 10"  -A 2 -B 1
*
00040800  bb f6 1e 00 00 00 00 00  31 6d 6f 74 65 2e 63 66  |........1mote.cf|
00040810  67 00 00 00 01 04 11 00  00 00 05 10 de ad be ef  |g...............|
00040820  de ad be ef de ad be ef  de ad be ef 03 04 39 05  |..............9.|
00040830  00 00 ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
--
*
00046000  b2 8a 1e 00 00 00 00 00  31 6d 6f 74 65 2e 63 66  |........1mote.cf|
00046010  67 00 00 00 01 04 0f 00  00 00 05 10 de ad be ef  |g...............|
00046020  de ad be ef de ad be ef  de ad be ef 03 04 39 05  |..............9.|
00046030  00 00 ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
--
*
00062000  f7 40 18 00 00 00 00 00  31 6d 6f 74 65 2e 63 66  |.@......1mote.cf|
00062010  67 00 00 00 03 04 39 05  00 00 05 10 00 11 22 33  |g.....9........"3|
00062020  44 55 66 77 88 99 00 11  22 33 44 55 ff ff ff ff  |DUfw....."3DU....|
00062030  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
--
*
00069800  f9 32 14 00 00 00 00 00  31 6d 6f 74 65 2e 63 66  |.2......1mote.cf|
00069810  67 00 00 00 05 10 00 11  22 33 44 55 66 77 88 99  |g......."3DUfw..|
00069820  00 11 22 33 44 55 03 04  ff ff ff ff ff ff ff ff  |.."3DU..........|
00069830  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
0006a000  ff 96 18 00 00 00 00 00  31 6d 6f 74 65 2e 63 66  |........1mote.cf|
0006a010  67 00 00 00 05 10 00 11  22 33 44 55 66 77 88 99  |g......."3DUfw..|
0006a020  00 11 22 33 44 55 03 04  39 05 00 00 ff ff ff ff  |.."3DU..9.......|
0006a030  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
0006b000  93 0e 18 00 00 00 00 00  31 6d 6f 74 65 2e 63 66  |........1mote.cf|
0006b010  67 00 00 00 05 10 69 69  69 69 69 69 69 69 69 69  |g.....iiiiiiiiii|
0006b020  69 69 69 69 69 69 03 04  39 05 00 00 ff ff ff ff  |iiiiii..9.......|
0006b030  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
0006b800  2b 73 18 00 00 00 00 00  31 6d 6f 74 65 2e 63 66  |+s......1mote.cf|
0006b810  67 00 00 00 05 10 69 69  69 69 69 69 69 69 69 69  |g.....iiiiiiiiii|
0006b820  69 69 69 69 69 69 03 04  39 05 00 00 ff ff ff ff  |iiiiii..9.......|
0006b830  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
```

Figure 5.10: Memory dump of a Linear Mote Eterna and several Join Keys used.

5.3 Linear Access Point Manager LTP5903CEN-WHR

This section describes how the firmware image from the Linear Manager *LTP5903CEN-WHR* was carried out and undocumented XML-RPC credentials. Although the Linear Manager was not our target, it is considered to mention. Furthermore, we will use its RF module to attempt its content and analyze it.

5.3.1 Firmware extraction

In order to extract the filesystem from the AP Manager several steps must be carried out in advance. Figure 5.11 shows the same instructions in order to dump the firmware image.

- Login in the AP Manager with administrator privileges. The default password for `root` is `sur+cycl3s`.
- Add an IPtable rule to accept all connections.
- Dump into a TCP socket the content of `mtdblock1` using `netcat` in server mode (listening).
- From a reachable machine in the network, connect and download the mtdblock image.

The AP Manager was already including the networking tool called `netcat`, therefore the firmware extraction was trivial. If the device had not `netcat`, it would be always possible to statically cross compile any version of `netcat` for the ARM architecture and subsequently upload it to the router over a SSH, HTTP or USB connection.

```
■ From the Linear shell with root privileges:
sh-3.2# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00400000 00004000 "Bootstrap"
mtd1: 03c00000 00004000 "Partition 1"
mtd2: 00000000 00000000 "Partition 2"
sh-3.2# iptables -A INPUT -s 0.0.0.0 -j ACCEPT
sh-3.2# cat /dev/mtdblock1 | nc -l -p 6666
sh-3.2# md5sum /dev/mtdblock1
baa0c3207dbcfad16ccc655946735a5c  mtdblock1

■ From a normal PC:
edu@ubuntu:/tmp$ nc -n -v 192.168.99.100 6666 > mtdblock1.bin
Connection to 192.168.99.100 6666 port [tcp/*] succeeded!
edu@ubuntu:/tmp$ md5sum mtdblock1.bin
baa0c3207dbcfad16ccc655946735a5c  mtdblock1.bin
```

Figure 5.11: Firmware extraction of the Linear Manager LTP5903CEN-WHR.

5.3.2 Credentials Found

After extracting the firmware image running in the AP Manager, a couple of suspicious users-passwords for a XML-RPC [6] service were found. This protocol encodes requests by using XML and uses HTTP as the transport mechanism. These passwords have been tested against the AP Manager without success. This finding was discovered almost at the end of the thesis has not been 100% double checked.

[6] http://en.wikipedia.org/wiki/XML-RPC

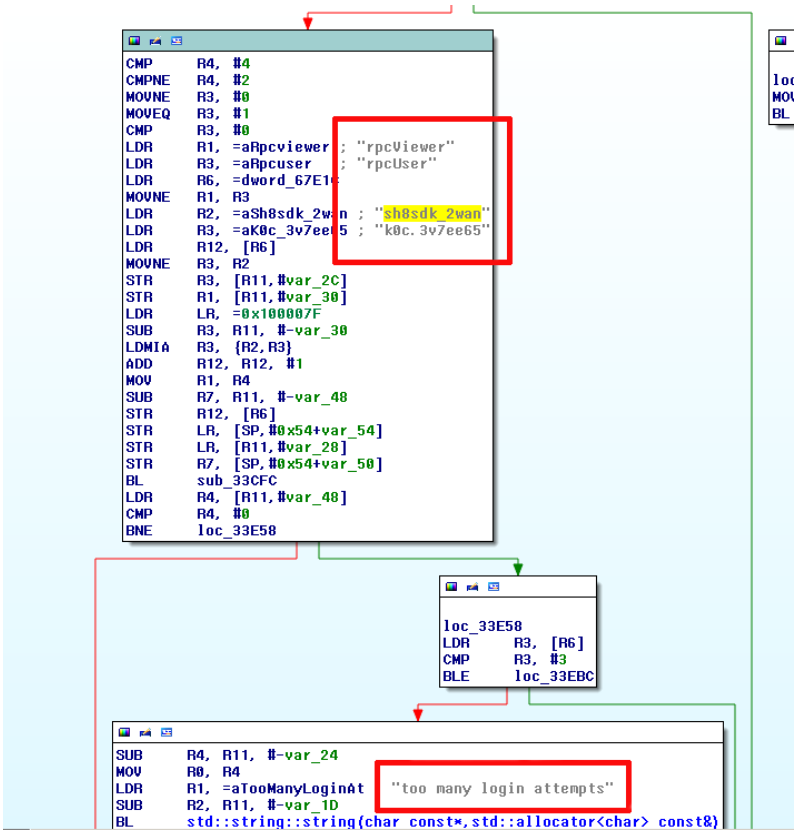


Figure 5.12: XML-RPC users and passwords not documented.

5.4 A WirelessHART `unknown-mote`

This section briefly addresses certain aspects of an `unknown-mote`. Once again, the main goal is to retrieve the cryptographic AES *JoinKey*. This section explains what was attempted, what failed, what tools were needed during the research and possible ideas for a future research on these devices will be addressed in the last chapter. The `unknown-mote` has a metallic shield as physical security. As usual, the RF module is protected with a Faraday cage. After opening up the device and verifying that the `unknown-mote` was still working, it was clear that there were no anti-tampering measures. Our first goal was to identify as many components as possible to start finding out where the *JoinKey* was stored.

5.4.1 Hardware Components: `unknown-mote`

Once opened up, there was an annoying epoxy layer to protect the PCB. Our first attempt was to identify the components on the device. Unlike the Linear mote, the `unknown-mote` has several MCUs which are interconnected with each other. The `unknown-mote` follows the specifications which were discussed in Chapter 2 with distinct MCUs for managing different tasks on the wireless sensor node.

The `unknown-mote` is comprised of the following components:

- `Main CPU`: AVR ATMEGA 8bits with internal flash. Fuses and lock bits.
- `Radio Frequency`: IEEE 802.15.4. DN2510 Dust Network. Package: BGA.
- `Sensor Module`: TI MSP430 MCU.

- `Hardware Crypto-Engine`: AES
- `EEPROM`: Store name and serial number of the wireless node. It can be seen covered up with a gray rubber in Figure 5.14.

Our first approach was to read documentation and consider which were the alternatives. Initially, the hardware reverse engineering part concluded with no much information. The RF chip had not any documentation online about its structure nor architecture. The TI chip was clearly confusing due to the tough visibility of its chip mark. A magnifier was necessary to continue figuring out more details about the `unknown-mote`. Although the main AVR SoC was clearly identified we did not hold an AVR JTAG programmer and possibly its hardware security should be enabled. Our initial approach was to start discarding candidates and leaves the AVR chip as a later option. A lot of literature WSNs was pointing to TI chips and we decided to invest in a JTAG programmer for such chips. GoodFET was our best match for our task. A hardware open source tool with a lot of community behind of the project.

5.5 Microcontroller TI MSP430

At the time of exploring this MCU it was neither known that it was a TI chip nor that it was a sensor module. Once the device was opened up, it was really tough to figure out the mark of this MCU. After a while, some microscopic pictures revealed the mark as a TI msp430 MCU. This information bumped into some research about how to extract cryptographic keys in wireless ZigBee radios [71, 72].

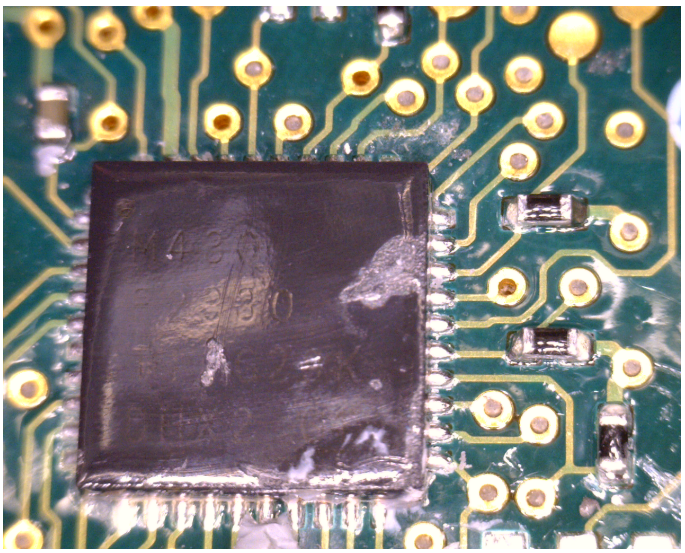


Figure 5.13: TI MSP430 MCU discovered by using a microscope.

In Figure 5.13 is a clear indication that we face against a TI MSP430. Figure 5.14 depicts the four mandatory cables in order to JTAG the MCU. Eventually, Figure 5.15 shows the device memory dumping by using a GoodFET JTAG programmer.

After reviewed the memory content of the sensor MCU is not revealing any conclusive data about the evidence of cryptographic keys in there. Reviewing the memory dump we did not find the the *JoinKey* which we knew in advance. A bit of reverse engineering was carried out by disassembling the memory content with Interactive Disassembler (IDA) Pro. To do so, it is important to specify the MSP430 MCU as main CPU to disassemble machine code. After loaded

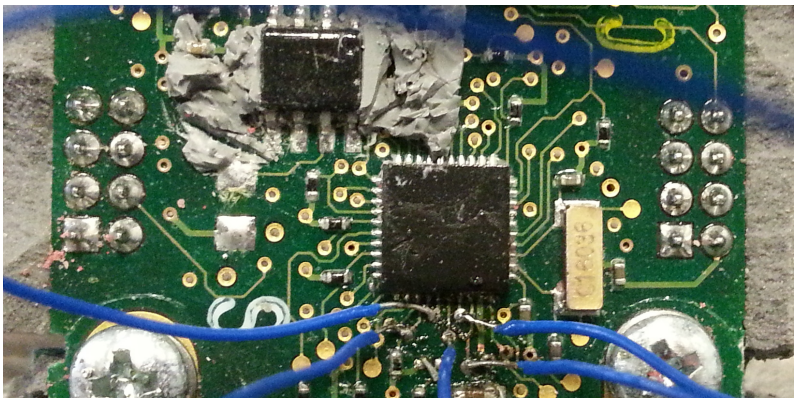


Figure 5.14: Hooking up cables in the JTAG pins in TI MSP430 MCU.

```
$ ~/goodfet/client$ sudo goodfet.msp430 info
Identifies as MSP430F23x0 (f237)


$ ~/goodfet/client$ sudo goodfet.msp430 dump msp430_flash_all.hex 0x0000 0xffff
Dumping from 0000 to ffff as msp430_flash_all.hex.
Dumped 000000.
Dumped 000400.
Dumped 000800.
......
Dumped 00f000.
Dumped 00f400.
Dumped 00f800.
Dumped 00fc00.

$ ~/goodfet/client$ sudo goodfet.msp430 ivt
ffc0 ffff
ffc2 ffff
ffc4 ffff
.....
ffe6 f8f8
....
fffa f76c
fffc ffff
fffe e006
```

Figure 5.15: JTAG'ing the TI MSP430 MCU with the GoodFET.

into IDA Pro, we were capable of obtaining cross references into the machine code. However, no-valuable information was found nor obfuscation of the internal memory as boot loaders normally do. Finally, we conclude that *JoinKeys* are not stored in TI chips and this chip is only used for sensor tasks.

5.6 AVR Microcontroller

This section describes how to read the fuses and lock bits for an AVR chip. The concrete model is not revealed but the process is commented. In order to read the security measures, an AVR programmer is needed to start communicating with the chip and extract the fuses and lock bits. As discussed before, the AVR Dragon is a low-cost offering from Atmel and it was our final choice.

In order to read the fuses and lock bits several tools were used:

- `AVRdude` and `AVaRICE`. These utilities are used for manipulating the ROM and EEPROM contents of AVR microcontrollers using the ICSP or JTAG. Both are command-line-based

tools and do not require much space in our Operating System (OS). Our strategy was firstly to read an AVR chip without security fuses and extract the fuses, lock bits and memory content. Once obtained, we performed several attempts against our real AVR target without any useful information. In conditions where fuses are disabled, the output of such programs is to show values for the fuses and lock bits. However, if we disable the JTAG and SPI interface on our controllable AVR, the device remains completely inaccessible. We can always clean fuses up but removing the memory content before resetting fuses. This does not help to go further. Figure 5.16 depicts attempts against the AVR target.

```
$ sudo avrdude -p mXYZ -P usb -c dragon_jtag -U lfuse:r:low:r -U hfuse:r:high:r -U efuse:r:ext:r -F

avrdude: jtagmkII_program_enable(): bad response to enter progmode command: RSP_ILLEGAL_JTAG_ID
avrdude: jtagmkII_program_enable(): bad response to enter progmode command: RSP_ILLEGAL_JTAG_ID
avrdude: JTAGEN fuse disabled?
avrdude: initialization failed, rc=-1
avrdude: AVR device initialized and ready to accept instructions
avrdude: Device signature = 0xc8974f
avrdude: Expected signature for ATmegaXYZ is XX YY ZZ

avrdude done.  Thank you.


$ sudo avrdude -p mXYZ -P usb -c dragon_jtag -U flash:r:AVR_dump.hex:r

avrdude: jtagmkII_program_enable(): bad response to enter progmode command: RSP_ILLEGAL_JTAG_ID
avrdude: jtagmkII_program_enable(): bad response to enter progmode command: RSP_ILLEGAL_JTAG_ID
avrdude: JTAGEN fuse disabled?
avrdude: initialization failed, rc=-1
         Double check connections and try again, or use -F to override
         this check.


avrdude done.  Thank you.
```

Figure 5.16: Trying to JTAG the AVR chip with an AVR Dragon.

- `Atmel Studio 6`. This the official tool which provides advanced programming and debugs connectivity for Atmel ARM- and AVR-based MCUs, including the ability to capture data trace information. Once created a project with our specific AVR type, we were unable to achieve the fuses and lock bits. Unfortunately, we did not get further information after this error:

  "**Unable to enter programming mode. JTAGID not valid. Debugger command enterProgMode failed**"

Concluding after various tests against the main MCU reveal that the both JTAG and SPI interfaces seem to be disabled.

Our hypothesis was that the *JoinKey* was not inside of the AVR chip because as it is known, the RF module is the only capable of encrypting and decrypting via hardware crypto-engine with proper performance. Therefore our last attempt was to sniff the SPI bus after booting the wireless sensor node to verify that the *JoinKey* was not residing inside of the AVR chip. We did not see any evidence of the *JoinKey* anywhere. Since we do not hold any set up for SCA attacks nor other more sophisticated equipment, we stopped at this point.

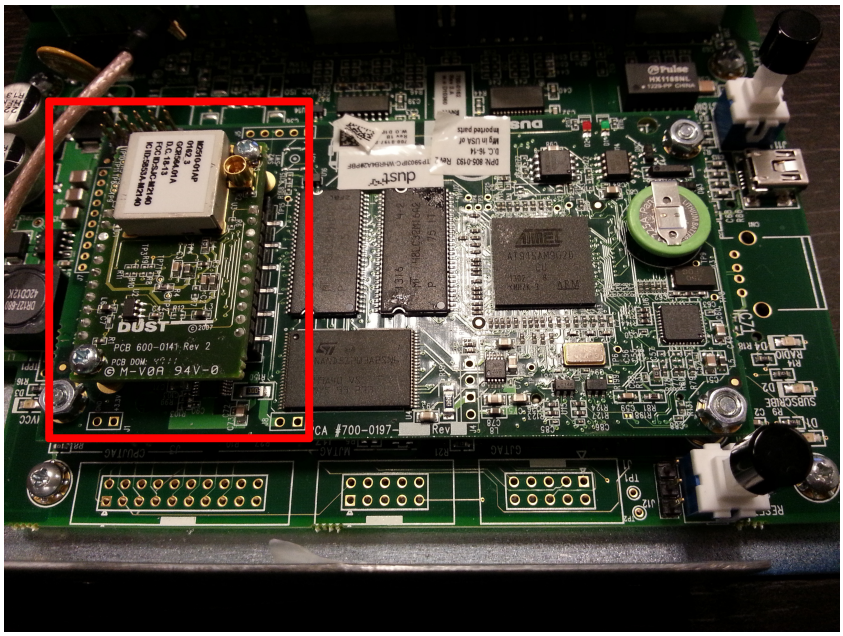


Figure 5.17: The internals of the Linear AP Manager and its M2510 highlighted with red colour.

5.7 The BGA chip: DN2510

Unfortunately, the RF chip is a BGA package without any public datasheet available on the Internet. The mark on the chip redirects us to its manufacturer: Dust Networks. After some months of research, we were able to access to the datasheet by constantly contacting with the manufacturer. Knowing the pinout of balls does not provide us much help either. Nevertheless, it was discovered that the Linear AP Manager had the same BGA chip for the radio purpose. Figure 5.17 shows the RF chip covered with a metallic shield. Fortunately, this RF module was offered in a socket which has pins and the pinout is kindly offered in its website. The fact of having the BGA socket with proper pinout allowed us to interact with the programming content.

To show the process carried out, in Figure 5.17 is shown the internals of the Linear AP Manager and its M2510 socket highlighted with red color also referenced in this section as Linear AP Manager RF chip.

Figure 5.18 depicts in number (1) the RF chip DN2510 in the `unknown-mote` and some possible pins. Furthermore, in (2) we appreciate the same RF chip but in the Linear AP Manager. Since we knew the pinout of the socket, we carried out trial and error with a multimeter to bruteforce the pins near the RF chip. As we were able to read and write the RF chip from the Linear Kit and we also knew the SPI pins, our idea was to bruteforce the test pads around the chip on the `unknown-mote`. We verified that in the Linear AP Manager some pins at the bottom of the socket, see Figure 5.19 to observe bottom pins, were producing conductivity between the top ones by using a multimeter. Top pins are highlighted with read lines in Figure 5.18. In order to read or write the RF chip via SPI, at least we must know 6 pins. GROUND is easy to detect, and we suppose that the 3 pins highlighted in Figure 5.18 are the same ones in both pictures in Figure 5.18. This leaves only 2 possible combinations. Connecting the 3 'possible' pins : SCK, MISO and MOSI as well as GND, it did not get any useful data. The way we verified if a possible combination worked, it consisted as follows: cables were connected and we used `flashrom` for reading the BGA. Finally, all readings were not finding any SPI memory and therefore no further reading. Unfortunately, such combinations did not produce any response to our SPI readings. Nevertheless, at the right side of the RF chip from the `unknown-mote` we can distinguish exactly

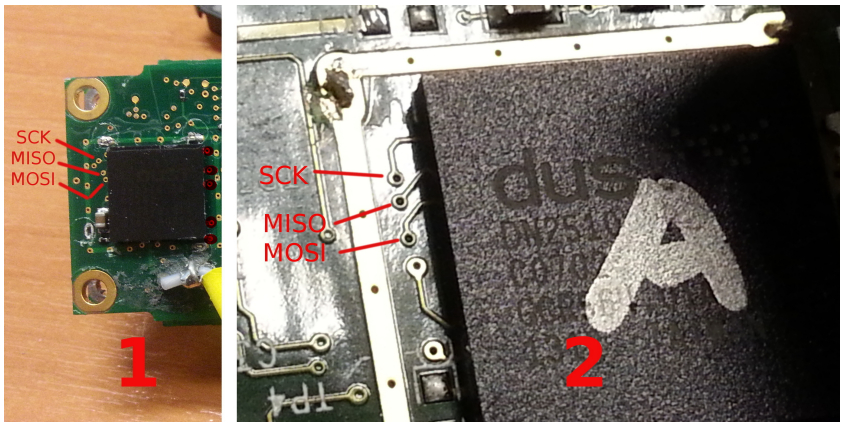


Figure 5.18: (1) The Radio chip DN2510 in the unknown-target and possible pins for SPI. (2) The Radio chip in the Linear AP Manager which is readable through a socket.

5 test pads highlighted with red circles. This attempt failed either because soldering problems or because pins were not matching properly. If we wanted to bruteforce 8 pins, the possible combinations are 8! or 40320. Trying these by hand is unfeasible. An interesting project would have been to develop a bruteforcer in an Arduino board. Similar projects are existing for finding out the pinout of UART [84] and JTAG [84, 85]. However, this was not possible during the research period.

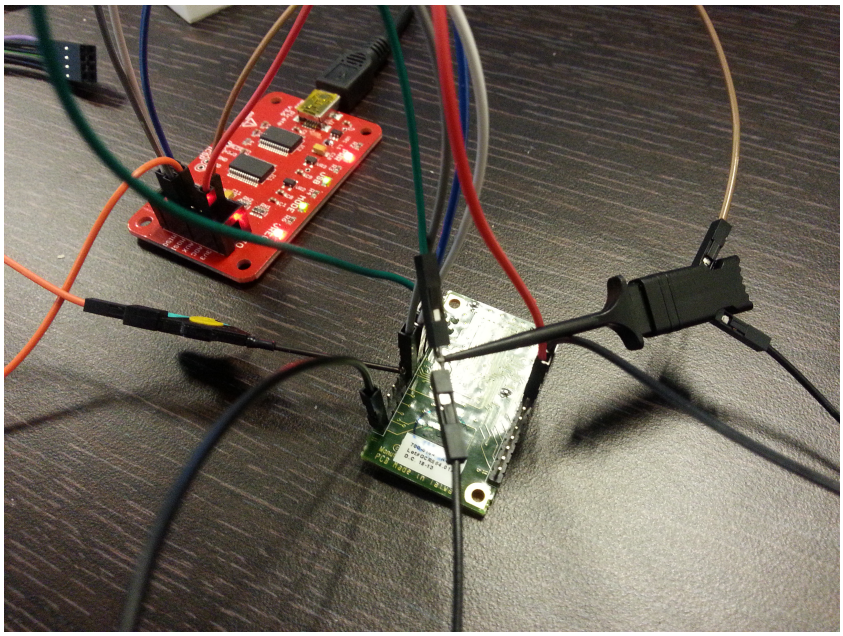


Figure 5.19: Dumping out the M2510 BGA memory through a socket and the Bus Pirate v3.6.

Although it has been shown how to extract the information from this BGA chip in a socket, we still find an issue. If we can suppose a possible attack where desoldering and soldering was possible. The idea would be to desolder the BGA chip from the `unknown-mote` and solder it into the socket. Subsequently, read all data out by using a Bus Pirate and the tool `flashrom`. The issue is that even when the BGA memory has been desoldered from the `unknown-mote`, and soldered back into the M2510 socket. Then, there are possibilities of that the distinct firmware image is running on the `unknown-mote` and the AP Manager. The Linear kit is an AP manager and its RF

chip is acting as 'server' or access point. However, the `unknown-mote` is running a software acting as 'client'. Furthermore, both use the same hardware wireless node, but they are not produced by the same re-seller and we cannot verify the same behavior. Eventually, it was tried that the Linear AP manager was able to maintain the *JoinKey* in a non-volatile way even without the RF chip. It was not found any *JoinKey* in the BGA memory in the AP Manager either. We conclude that this BGA target cannot be faced without proper tools and more experience with hacking embedded devices which use BGA packages.

```
$ hd mote_M2510_testCS_2.bin | more
00000000  00 00 00 c4 00 44 4e 00  00 92 b4 d3 25 db 3a 4d  |.....DN.....%.:M|
00000010  08 2d 25 ab ef 7e 05 18  c4 52 1a 1a dc 87 64 0a  |.-%..~...R....d.|
00000020  e1 71 ad 5c 52 48 5e 60  b4 1d 61 68 b1 d2 d4 d1  |.q.xxxxx..ah....|
00000030  46 9f e3 12 8a 6b 4d f5  f1 e1 ae cd 3f 0f ed aa  |F....kM.....?...|
00000040  d7 46 d9 1e a6 4d 4a 78  d0 dc 3e 8f 0a 27 80 7a  |.F...MJx..>..'.z|
00000050  43 7a 9f b1 ea f3 61 c3  8c 2d 98 9d 0c 17 70 9b  |Cz....a..-....p.|
00000060  3d bc 48 19 4b 3d dd 8f  d8 e5 ae 56 78 ec fd 8b  |=.H.K=.....Vx...|
00000070  7c f2 e5 f5 f5 10 fe dd  7e 2e 88 83 fe a9 7d 83  |e....... .....o.|
00000080  68 1c 17 43 16 74 c5 59  56 c7 24 e1 4f 08 9f bd  |h..C.t.YV..O....|
00000090  a6 e8 f2 ec 70 f1 03 5c  01 02 25 e9 d8 3c 1e b1  |....p.. e d..<..|
000000a0  44 4d 06 0f a1 92 d6 f7  ad c4 11 0b b2 1e 4a de  |DM.............J.|
000000b0  2c 9f d8 5f 85 fb 29 d5  56 37 79 64 b3 d1 7f d8  |,.._..).V7yd....|
000000c0  be c0 2f 97 7a 4e f4 97  2a fb 70 ff ff ff ff ff  |../.zN..*.p.....|
000000d0  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
```

Figure 5.20: Dumping the Linear DN2510 BGA memory by using a BusPirate v3.6.

Furthermore, in Figure 5.20 is shown what it was dumped in the Linear M2510 BGA memory by using a BusPirate v3.6 and how it was carried out in Figure 5.19. Figure 5.20 shows the header of the memory dump from a RF chip in the Linear AP Manager. A clear evidence of a proper reading depicts the letters "`DN`" at the first offsets.

Chapter 6

Conclusion

6.1 Conclusions

Taking into consideration different hurdles and successful ideas during the way of this research. I would like to conclude that:

- During this research, several low-cost physical attacks have been carried out in WSNs. In our first development board, we concluded recovering its cryptographic keys using hardware techniques. In our second target was not possible to extract cryptographic keys on time and some ideas remained for future work. However, we conclude that all the wireless sensor nodes held their cryptographic keys inside of the on-board memory. We found out that either the cryptographic keys were stored in a protected area or in the memory of the radio chip without security measures. In the latter case, we were able to measure how difficult a BGA chip could be read without proper equipment.

- The WirelessHART key management and joining process were addressed through diagrams and explanations step-by-step to make easier their clear understanding. Some aspects remained as unknown due to the no clarity and specification lacking of the WirelessHART standard at the time of writing. We conclude that do not know how the session key derivation phase is internally done and there is no clear statement how the information is ensured between the following WirelessHART components: Network Manager, Access Point and Gateway. It is responsibility of certain companies the developing of such critical operations.

- We provided a journey through of the different networking layers and discussed the security measures as well as some details about the structure of packages. We did not go deeper in this topic due to our goals were different ones. However, security measures were discussed at different IEEE 802.15.4 layers and the WirelessHART standard was compared with similar wireless technologies such as *Zigbee* or *ISA 100* in order to get an impression about their performance and security level.

- Despite the fact that the certain information about the physical specifications was not obtained until the end of the research, the approach was always to learn about hardware techniques and architectures and estimate how much work would result for a motivated attacker. We noticed how black-box projects can result tough and time-consuming tasks. However, we believe that both wireless nodes could be improved regarding security measures.

- Initially there exists a gap between the software people who start developing hardware skills. The learning curve might be shorter or larger depending on the target. This research taught me various perspectives and challenges. In black-box reverse engineering projects, we totally do not know how the project can change from one moment to another.

- It is known that many manufacturers enforce the use of BGA chips in hardware designs to stop low-cost hacking techniques. Another technique is cover the flash's tracks and pads with glue to disturb tampering or even bury them between the PCB layers. The BGA chip totally impeded further advance in the research of this topic. First, because the author had neither the tools nor the knowledge to carry out a BGA rework technique. It this had been possible, it could have been dealt from different perspectives. Unfortunately, the author of this research did not have background on how to reball BGA chips. Furthermore, to buy a specific socket for the programmer can be costly for a simple reading. Prices are between €350 and €500 for a specific BGA socket for an EEPROM reader. Something that was thought during the research was to outsource to another company to carry out this task. Unfortunately, we were not completely sure that the *JoinKey* would be readable in there. It is important to remark that we were able to dump the content of another BGA chip since this chip was in a socket already prepared. After reading we did not have any clear conclusion of where the *JoinKey* was stored. This might happen because such BGA chips were running different firmware images. Whereas the mote was running client software, the AP Manager was running server software. Our first target was storing the *JoinKey* in the flash without any obfuscation nor encryption.

6.2 Further Work

After writing this thesis some ideas could be carried out in future research on similar WirelessHART devices:

- Unfortunately, at the time of writing I have not been able to use any laboratory and special equipment for SCA and suck attacks are out of the scope of this thesis. When a chip is protected using fuses, an attacker with non-expensive equipment can carry out half of the attacks with a proper one. With this, I want to remark that hardware attacks are very equipment-dependent and although depending on the knowledge of anyone, an attacker can be stopped by the lack of this exact equipment. Unlike software-based projects, this problem is less visible in software

- The `unknown-mote`'s *JoinKey* is stored in the non-volatile flash memory in the RF chip. Either obtain a specific BGA socket for the EEPROM reader in order to read out the flash memory or try to keep bruteforcing some external pins to figure out which pins allow to interact with it. According to its datasheet, neither fuses nor lock bits are used. JTAG is not supported, leaving the SPI interface as the only programming interface in the chip.

- Since it was proven that the memory content can be extracted from the socket M2510, which has the same BGA chip without any hardware security measures then it could result interesting to try to desolder the BGA from the `unknown-mote` and solder it into the M2510 socket. This would allow us to find out whether either really the *JoinKey* is in there or not. A unsolved question is if different firmware images deal the secret key storage in different manner. Mainly because the `unknown-mote` runs a 'client' node and the Linear AP Manager runs a 'server' node or manager mode.

- Although we recovered the *JoinKey* for the Linear WirelessHART Kit Starter, it would be really interesting to set the fuses and attempt different and techniques in order to extract the cryptographic key. If this idea is carried out, a more sophisticated laboratory must be also set up.

- The WirelessHART standard is not well defined yet regarding its session keys generation. It is believed to be secure because using AES-128 bits in CBC-MAC mode. However, neither we know how the PRNG is internally implemented in devices nor where the entropy is coming from. Along the document several cases have been shown where keys were generated by using the date as seed for the PRNG. An interesting research could be focused on this topic.

The main problem found is the lack of information for certain points of the WirelessHART standard. Even the contactless smartcards most widely used nowadays seem to mess up this very important topic as well. Further research would be an interesting and challenging future research.

- An interesting idea is to bring these devices to a SCA laboratory and try to extract the cryptographic keys by Differential Power Analysis techniques. For this set-up, we should power up the mote and take over the control of sending commands to the RF chip. Sending random payloads to be encrypted would produce different power consumption. This would allow us to apply statistical techniques to precompute the 16-bytes long for the AES-128 cipher. Controlling the trigger to measure just during the encryption process also sets out some challenging task. To conclude, these devices do not seem to be ready to be resilient against SCA attacks. Although no SCA attacks have been carried out, these targets might not have countermeasures against such attacks.

Bibliography

[1] Shahid Raza. Secure communication in wirelesshart and its integration with legacy hart. 2010. ixix, 28, 32, 33, 36, 45

[2] D. Puccinelli and M. Haenggi. Wireless sensor networks: applications and challenges of ubiquitous sensing. *IEEE Circuits and Systems Magazine*, 5(3):19–31, 2005. 1

[3] Marie Chan, Daniel Estève, Christophe Escriba, and Eric Campo. A review of smart homes- present state and future challenges. *Computer methods and programs in biomedicine*, 91(1):55–81, July 2008. 5

[4] Chris R Baker, Kenneth Armijo, Simon Belka, Merwan Benhabib, Vikas Bhargava, Nathan Burkhart, Artin Der Minassians, Gunes Dervisoglu, Lilia Gutnik, M Brent Haick, et al. Wireless sensor networks for home health care. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 2, pages 832–837. IEEE, 2007. 5

[5] BF Spencer. A study on building risk monitoring using wireless sensor network mica mote. In *First International Conference on Structural Health Monitoring and Intelligent Infrastructure, Japan*, pages 353–363, 2003. 6

[6] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24. ACM, 2004. 6

[7] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 254–263. IEEE, 2007. 6

[8] Shamim N Pakzad, Gregory L Fenves, Sukun Kim, and David E Culler. Design and implementation of scalable wireless sensor network for structural monitoring. *Journal of Infrastructure Systems*, 14(1):89–101, 2008. 6

[9] Hongki Jo, Sung-Han Sim, Kirill A Mechitov, Robin Kim, Jian Li, Parya Moinzadeh, BF Spencer Jr, Jong Woong Park, Soojin Cho, Hyung-Jo Jung, et al. Hybrid wireless smart sensor network for full-scale structural health monitoring of a cable-stayed bridge. In *Proceedings of the SPIE Smart Structures/NDE Conference*, 2011. 6

[10] Nicholas Brigman. Structural Health Monitoring in Commercial Aviation. 2012. 7

[11] MJ Chae, HS Yoo, JY Kim, and MY Cho. Development of a wireless sensor network system for suspension bridge health monitoring. *Automation in Construction*, 21:237–252, 2012. 7

[12] S Longhi, D Marzioni, E Alidori, G Di Buo, M Prist, M Grisostomi, and M Pirro. A Wireless Sensor Network Architecture for Solid Waste Management. pages 1–4. 7

[13] Narendra Kumar G, Chandrika Swamy, and K N Nagadarshini. Efficient Garbage Disposal Management in Metropolitan Cities Using VANETs. 2(3), 2014. 7

[14] Jatuporn Chinrungrueng, Udomporn Sunantachaikul, and Satien Triamlumlerd. Smart Parking: An Application of Optical Wireless Sensor Network. *2007 International Symposium on Applications and the Internet Workshops*, pages 66–66, 2007. 8

[15] Manjusha Patil and Vasant N Bhonge. Wireless Sensor Network and RFID for Smart Parking System. *International Journal of Emerging Technology and Advanced Engineering*, 3(4):188–192, 2013. 8

[16] Satish V Reve and Sonal Choudhri. Management of Car Parking System Using Wireless Sensor Network. 2(7):262–268, 2012. 8

[17] Mohamed Amine Kafi, Yacine Challal, Djamel Djenouri, Messaoud Doudou, Abdelmadjid Bouabdallah, and Nadjib Badache. A Study of Wireless Sensor Networks for Urban Traffic Monitoring: Applications and Architectures. *Procedia Computer Science*, 19:617–626, 2013. 8

[18] Wireless sensor networks for environmental noise monitoring. *6th GI/ITG Workshop on Sensor Networks*, pages 1–4, 2007. 8

[19] Eiman Kanjo. Noisespy: A real-time mobile phone platform for urban noise monitoring and mapping. *Mobile Networks and Applications*, 15(4):562–574, 2010. 8

[20] Jayavardhana Gubbi, Slaven Marusic, Aravinda S. Rao, Yee Wei Law, and Marimuthu Palaniswami. A pilot study of urban noise monitoring architecture using wireless sensor networks. *Proceedings of the 2013 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2013*, (1999):1047–1052, 2013. 8

[21] Kavi K Khedo, Rajiv Perseedoss, Avinash Mungur, University Of Mauritius, and Mauritius. A Wireless Sensor Network Air Pollution Monitoring System. *Science*, 2:15, 2010. 9

[22] Raja Vara Prasad Y, Mirza Sami Baig, Rahul K Mishra, P Rajalakshmi, U B Desai, and S N Merchant. Real Time Wireless Air Pollution Monitoring System. *Ictact Journal on Communication Technology: Special Issue on Next Generation Wireless Networks and Applications*, 2:370–375. 9

[23] Vasim K Ustad and Suhas S Kibile. Zigbee Based Wireless Air Pollution Monitoring System Using Low Cost. 10(9):456–460, 2014. 9

[24] Godbless Swagarya, Shubi Kaijage, and Ramadhani S Sinde. A Survey On Wireless Sensor Networks For Air Pollution Monitoring. 3(5):5975–5979, 2014. 9

[25] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications - WSNA '02*, page 88, 2002. 9

[26] Tomasz Naumowicz, Robin Freeman, Holly Kirk, Ben Dean, Martin Calsyn, Achim Liers, Alexander Braendle, Tim Guilford, and Jochen H. Schiller. Wireless sensor network for habitat monitoring on skomer island. In *The 35th Annual IEEE Conference on Local Computer Networks, LCN 2010, 10-14 October 2010, Denver, Colorado, USA, Proceedings*, pages 882–889, 2010. 9

[27] Philipp M Glatz, Christian Steger, and Reinhold Weiss. A wireless sensor node for river monitoring using MSP430 and energy harvesting. 10

[28] Lívia C Degrossi, Guilherme G Amaral, Eduardo S. M. De Vasconcelos, João Porto De Albuquerque, and Jó Ueyama. Using Wireless Sensor Networks in the Sensor Web for Flood Monitoring in Brazil. *Proceedings of the 10th International ISCRAM Conference Baden-Baden, Germany, May 2013*, (May):458–462, 2013. 10

[29] Kwang-soo Kim and Tae-wook Heo. A Smart Grid Testbed using Wireless Sensor Networks in a Building. *SENSORCOMM 2011 : The Fifth International Conference on Sensor Technologies and Applications*, (c):371–374, 2011. 10

[30] Andrey Somov, Alexander Baranov, Alexey Savkin, Denis Spirjakin, Andrey Spirjakin, and Roberto Passerone. Development of wireless sensor network for combustible gas monitoring. *Sensors and Actuators A: Physical*, 171(2):398–405, 2011. 10

[31] Oscar Rorato, Silvano Bertoldo, Claudio Lucianaz, Marco Allegretti, and Riccardo Notarpietro. An Ad-Hoc Low Cost Wireless Sensor Network for Smart Gas Metering. 2013(March):61–66, 2013. 10

[32] Tariq Al-Kadi, Ziyad Al-Tuwaijri, and Abdullah Al-Omran. Wireless sensor networks for leakage detection in underground pipelines: A survey paper. *Procedia Computer Science*, 21:491–498, 2013. 10

[33] Sidra Rashid, Saad Qaisar, Husnain Saeed, and Emad Felemban. A Method for Distributed Pipeline Burst and Leakage Detection in Wireless Sensor Networks Using Transform Analysis. 2014, 2014. 11

[34] Imad Jawhar, Nader Mohamed, Khaled Shuaib, and Al Ain. A Framework for Pipeline Infrastructure Monitoring Using Wireless Sensor Networks. *Measurement*, pages 1–7, 2006. 11

[35] I. Stoianov, L. Nachman, S. Madden, T. Tokmouline, and M. Csail. PIPENET: A Wireless Sensor Network for Pipeline Monitoring. *2007 6th International Symposium on Information Processing in Sensor Networks*, pages 264–273, 2007. 11

[36] Nader Mohamed and Imad Jawhar. A Fault Tolerant Wired / Wireless Sensor Network Architecture for Monitoring Pipeline Infrastructures Nader Mohamed and Imad Jawhar. (August):186–191, 2008. 11

[37] M Lin and Y Wu. Wireless sensor network: Water distribution monitoring system. *Radio and Wireless Symposium, 2008 IEEE*, pages 775 – 778, 2008. 11

[38] Sunhee Yoon, Wei Ye, John Heidemann, Brian Littlefield, and Cyrus Shahabi. SWATS: Wireless sensor networks for steamflood and waterflood pipeline monitoring. *IEEE Network*, 25:50–56, 2011. 11

[39] Gyula Simon, Miklós Maróti, Ákos Lédeczi, György Balogh, Branislav Kusy, András Nádas, Gábor Pap, János Sallai, and Ken Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12. ACM, 2004. 11

[40] Tian He, Sudha Krishnamurthy, John A Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Jonathan Hui, and Bruce Krogh. Energy-efficient surveillance system using wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 270–283. ACM, 2004. 11

[41] Anish Arora, Prabal Dutta, Sandip Bapat, Vinod Kulathumani, Hongwei Zhang, Vinayak Naik, Vineet Mittal, Hui Cao, Murat Demirbas, Mohamed Gouda, et al. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, 2004. 11

[42] Tatiana Bokareva, Wen Hu, Salil Kanhere, Branko Ristic, Neil Gordon, Travis Bessell, Mark Rutten, and Sanjay Jha. Wireless sensor networks for battlefield surveillance. In *Proceedings of the land warfare conference*, 2006. 11

[43] Sang Hyuk Lee, Soobin Lee, Heecheol Song, and Hwang Soo Lee. Wireless sensor network design for tactical military applications: Remote large-scale environments. In *Proceedings of the 28th IEEE Conference on Military Communications*, MILCOM'09, pages 911–917, Piscataway, NJ, USA, 2009. IEEE Press. 12

[44] Byungrak Son, Yong-sork Her, and J Kim. A design and implementation of forest-fires surveillance system based on wireless sensor networks for south korea mountains. *International Journal of Computer Science and Network Security (IJCSNS)*, 6(9):124–130, 2006. 12

[45] Geoffrey Werner-Allen, Konrad Lorincz, Mario Ruiz, Omar Marcillo, Jeff Johnson, Jonathan Lees, and Matt Welsh. Deploying a wireless sensor network on an active volcano. *Internet Computing, IEEE*, 10(2):18–25, 2006. 12

[46] Geoffrey Werner-Allen, Jeff Johnson, Mario Ruiz, Jonathan Lees, and Matt Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 108–120. IEEE, 2005. 12

[47] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 381–396. USENIX Association, 2006. 12

[48] Pei Zhang, Christopher M. Sadler, Stephen a. Lyon, and Margaret Martonosi. Hardware design experiences in ZebraNet. *Proceedings of the 2nd international conference on Embedded networked sensor systems - SenSys '04*, page 227, 2004. 12

[49] Pavan Sikka, Peter Corke, and Leslie Overs. Wireless sensor devices for animal tracking and control. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 446–454. IEEE, 2004. 12

[50] Tim Wark, Peter Corke, Pavan Sikka, Lasse Klingbeil, Ying Guo, Chris Crossman, Philip Valencia, Dave Swain, and Greg Bishop-Hurley. Transforming agriculture through pervasive wireless sensor networks. *Pervasive Computing, IEEE*, 6(2):50–57, 2007. 13

[51] HART Communication Foundation LIT 34. HART Field Communication Protocol. 1999. 16, 20

[52] Electron18. Bell 202 Interface specification for transmission of binary data by frequency manipulation FSK. 2010. 16, 17

[53] White Paper of Digi.com. Demystifying 802.15.4 and ZigBee. 2008. 23

[54] Naagesh S Bhat. Design and implementation of ieee 802.15. 4 mac protocol on fpga. *arXiv preprint arXiv:1203.2167*, 2012. 25

[55] Juan Héctor Sánchez. Master thesis: Wirelesshart network manager. 2011. 27, 36

[56] Russell Housley. Using advanced encryption standard (aes) counter mode with ipsec encapsulating security payload (esp). 2004. 34

[57] Helger Lipmaa, David Wagner, and Phillip Rogaway. Comments to nist concerning aes modes of operation: Ctr-mode encryption. 2000. 34

[58] Morris Dworkin. *Recommendation for block cipher modes of operation: The CMAC mode for authentication*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2005. 35

[59] Junhyuk Song, Radha Poovendran, Jicheol Lee, and Tetsu Iwata. The aes-cmac algorithm. *RFC4493, IETF (June 2006)*, 2006. 35

[60] Vigil Security N. Ferguson MacFergus Hifn, R. Housley. Counter with cbc-mac (ccm). 35

[61] Shahid Raza, Thiemo Voigt, Adriaan Slabbert, and Krister Landernas. Design and implementation of a security manager for wirelesshart networks. In *Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on*, pages 995–1004. IEEE, 2009. 36, 40

[62] Shahid Raza, Adriaan Slabbert, Thiemo Voigt, and Krister Landernas. Security considerations for the wirelesshart protocol. In *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pages 1–8. IEEE, 2009. 45

[63] Shahid Raza, Gianluca Dini, Thiemo Voigt, and Mikael Gidlund. Secure key renewal in wirelesshart. *Real-time Wireless for Industrial Applications (RealWin11)-CPS Week*, 2011. 45

[64] Tomas Lennvall, Stefan Svensson, and Fredrik Hekland. A comparison of wirelesshart and zigbee for industrial applications. In *IEEE International Workshop on Factory Communication Systems*, volume 2008, pages 85–88, 2008. 45

[65] Mark Nixon and TX Round Rock. A comparison of wirelesshart and isa100. 11a. *Whitepaper, Emerson Process Management*, 2012. 46

[66] Gengyun Wang. Comparison and evaluation of industrial wireless sensor network standards isa100. 11a and wirelesshart. 2011. 46

[67] Travis Goodspeed. A side-channel timing attack of the msp430 bsl. *Black Hat USA*, 2008. 53

[68] Travis Goodspeed. Practical attacks against the msp430 bsl. 2008. 53

[69] Braden Thomas. Reverse Engineering the Supra iBox. *BlackHat USA*, 2014. 53

[70] Carl Hartung, James Balasalle, and Richard Han. Node compromise in sensor networks: The need for secure systems. *Department of Computer Science University of Colorado at Boulder*, 2005. 53

[71] Travis Goodspeed. A side-channel timing attack of the msp430 bsl. *Black Hat USA*, 2008. 53, 71

[72] Travis Goodspeed. Practical attacks against the msp430 bsl. In *Twenty-Fifth Chaos Communications Congress. Berlin, Germany*, 2008. 53, 71

[73] Travis Goodspeed. Reversing and exploiting wireless sensors. *Arlington, VA, February*, 2009. 53

[74] Lucas Apa and Carlos Mario Penagos Hollman. Compromising industrial facilities from 40 miles away. *IOActive Technical White Paper*, 2013. 53

[75] Giacomo De Meulenaer and François-Xavier Standaert. Stealthy compromise of wireless sensor nodes with power analysis attacks. In *Mobile Lightweight Wireless Systems*, pages 229–242. Springer, 2010. 54

[76] Kanthakumar Pongaliur, Zubin Abraham, Alex X Liu, Li Xiao, and Leo Kempel. Securing sensor nodes against side channel attacks. In *High Assurance Systems Engineering Symposium, 2008. HASE 2008. 11th IEEE*, pages 353–361. IEEE, 2008. 54

[77] Zoya Dyka and Peter Langendörfer. Improving the security of wireless sensor networks by protecting the sensor nodes against side channel attacks. In *Wireless Networks and Security*, pages 303–328. Springer, 2013. 54

[78] Edith CH Ngai, Jiangchuan Liu, and Michael R Lyu. An efficient intruder detection algorithm against sinkhole attacks in wireless sensor networks. *Computer Communications*, 30(11):2353–2364, 2007. 54

[79] Vinay Soni, Pratik Modi, and Vishvash Chaudhri. Detecting sinkhole attack in wireless sensor network. *International Journal of Application or Innovation in Engineering & Management*, 2(2), 2013. 54

[80] Murad A Rassam, Anazida Zainal, Mohd Aizaini Maarof, and Mohammed Al-Shaboti. A sinkhole attack detection scheme in mintroute wireless sensor networks. In *Telecommunication Technologies (ISTT), 2012 International Symposium on*, pages 71–75. IEEE, 2012. 54

[81] Issa Khalil, Saurabh Bagchi, and Ness B Shroff. Mobiworp: Mitigation of the wormhole attack in mobile multihop wireless networks. *Ad Hoc Networks*, 6(3):344–362, 2008. 54

[82] Zaw Tun and Aung Htein Maw. Wormhole attack detection in wireless sensor networks. *World Academy of Science, Engineering and Technology*, 46:2008, 2008. 54

[83] Dust Networks and Linear. Dc9007a - smartmesh wirelesshart starter kit. 2013. 61

[84] Nathan Fain. Jtagenum. *27c3: JTAG/Serial/FLASH/PCB Embedded Reverse Engineering Tools and Techniques*, 2010. 75

[85] Joe Grand. Jtagulator. *Grand Idea Studio*, 2013. 75