On the Impact of Energy Harvesting on Wireless Sensor Network Security

Di Mauro, Alessio; Dragoni, Nicola

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

On the Impact of Energy Harvesting on Wireless Sensor Network Security

A DISSERTATION PRESENTED

BY

ALESSIO DI MAURO

SEPTEMBER 2014

On the Impact of Energy Harvesting on Wireless Sensor Network Security

A DISSERTATION PRESENTED
BY
ALESSIO DI MAURO
TO
THE DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTER SCIENCE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
COMPUTER SCIENCE
PHD–2014–349

TECHNICAL UNIVERSITY OF DENMARK
KONGENS LYNGBY, DENMARK
SEPTEMBER 2014

Thesis advisor: Nicola Dragoni Alessio Di Mauro

On the Impact of Energy Harvesting on Wireless Sensor Network Security

Abstract

Given the continuous advancements in the technology of energy harvesting over the last few years, we are now starting to see wireless sensor networks (WSNs) powered by scavenged energy. This change in paradigm has major repercussions not only on the hardware engineering aspects, but also on the software side. The first protocols specifically designed to take advantage of the energy harvesting capabilities of a network have just recently appeared. At the same time, security remains one of the central points of WSNs development, because of their intrinsically unreliable nature that combines a readily accessible communication infrastructure such as wireless data exchange, to an often likewise readily accessible physical deployment. This dissertation provides a comprehensive look at how security can be improved by what energy harvesting has to offer. The main question asked is whether or not it is possible to provide better security in a WSN, by being aware of the fact that the amount of available energy is not going to monotonically decrease over time. The work covers different aspects and components of a WSN and focuses on what is arguably one the most important ones, medium access control (MAC) protocols. An energy-harvesting specific MAC protocol is introduced together with a related security suite. A new attack relevant to a whole class of MAC protocols is also introduced, along with a scheme that defeats it. A security approach for MAC protocols is discussed to provide an energy-aware solution. In

order to address security bootstrapping, a new energy-adaptive key reinforcement scheme is presented. Finally an implementation and some experimental results are provided.

Thesis advisor: Nicola Dragoni Alessio Di Mauro

On the Impact of Energy Harvesting on Wireless Sensor Network Security

Resumé

De seneste års fortsatte teknologiske fremskridt inden for energi-høst (energy harvesting) har ført til fremkomsten af trådløse sensor netværk (WSN) baseret på udnyttelse af indhøstet energi. Dette paradigmeskift har omfattende konsekvenser ikke alene for udvikling af hardware, men også for udvikling af software. For nyligt er fremkommet de første protokoller til netværk specielt udviklede med henblik på at udnytte energihøst. Et WSN, der kombinerer en let tilgængelig kommunikationsinfrastruktur, eksempelvis trådløs dataoverførsel, med en let tilgængelig fysisk implementering, er som udgangspunkt ikke pålideligt, hvorfor sikkerhed forbliver et centralt spørgsmål i udvikling af WSN. Denne afhandling giver en grundig analyse af, hvordan sikkerhed kan forbedres i et netværk baseret på energi-høst.

Et centralt spørgsmål er, hvorvidt der kan opnås en bedre sikkerhed i et WSN ved at erkende det forhold, at mængden af tilgængelig energi ikke er monotont faldende over tid. Arbejdet behandler forskellige aspekter af og komponenter benyttet i et WSN med fokus på MAC (medium access control) protokoller. En MAC protokol udviklet med henblik på energi-høst introduceres med tilhørende sikkerhedspakke. Et nyt angreb relevant for en klasse af MAC protokoller indføres, og en metode til bekæmpelse af sådanne angreb beskrives. En sikkerheds-adaptiv metodik diskuteres for MAC protokoller med henblik på at levere energibevidst sikkerhed, og en ny energi-adaptiv nøglebaseret metode for sikkerhed bliver foreslået. Endelig præsenteres

en implementering og eksperimentelle resultater.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AB altruistic backoff

ABR altruistic backoff request

AC alternating current

ACK acknowledgment

ACLK auxiliary clock

ADC analog to digital converter

AE authenticated encryption

AES advanced encryption standard

AM active mode

AnB Alice and Bob

AVISPA automated validation of Internet security protocols and applications

BFS breadth first search

BS base station

CBC cipher block chaining

CBC-MAC cipher block chaining message authentication code

CCA clear channel assessment

CMAC cryptographic message authentication code

IND-CPA indistinguishability under chosen plain-text attack

CPU central processing unit

CSMA carrier sense multiple access

DC duty-cycle

DC direct current

DCO digitally-controlled oscillator

DoS denial of service

ECO energy consumption overhead

EH energy harvesting

EH-WSN energy-harvesting wireless sensor network

ENO energy neutral operation

ENO-Max maximum performance energy neutral operation

GCM Galois counter mode

GE gate equivalent

IC integrated circuit

IF intermediate format

ISO international organization for standardization

IV initialization vector

LAR layer-based anycast routing

LPM low-power mode

MAC medium access control

MCLK main clock

MCU microcontroller unit

MPU microprocessor unit

NIST national institute for standards and technology

OCB offset codebook

ODMAC on-demand medium access control

OFMC on-the-fly model checker

OSI open systems interconnection

PDoS path denial of service

PSU power supply unit

QoS quality of service

RAM random access memory

RAP receiver authentication protocol

RAP-D receiver authentication protocol detection mode

RAP-P receiver authentication protocol prevention mode

RB random backoff

RF radio frequency

RI receiver-initiated

ROM read-only memory

RSSI received signal strength indicator

SI sender-initiated

SMCLK sub-main clock

SNEP secure network encryption protocol

SOC state of charge

USB universal serial bus

μTESLA micro timed efficient stream loss-tolerant authentication

VLO very-low-power low-frequency oscillator

WSN wireless sensor network

List of Relevant Papers

The following publications were the result of the research work presented and described in this dissertation:

[17] : Alessio Di Mauro, Davide Papini, and Nicola Dragoni. Security Challenges for Energy-Harvesting Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems (PECCS)*, pages 422–425. SciTePress, 2012

[18] : Alessio Di Mauro, Davide Papini, Vigo Roberto, and Nicola Dragoni. Introducing the Cyber-Physical Attacker to Energy-Harvesting Wireless Sensor Networks. *Journal of Networking Technology*, 3:139–148, 2012

[19] : Alessio Di Mauro, Davide Papini, Roberto Vigo, and Nicola Dragoni. Toward a Threat Model for Energy-Harvesting Wireless Sensor Networks. In *Proceedings of the 4th International Conference on Networked Digital Technologies (NDT)*, pages 289–301. Springer, 2012

[20] : Alessio Di Mauro, Xenofon Fafoutis, Sebastian Mödersheim, and Dragoni Nicola. Detecting and Preventing Beacon Replay Attacks in Receiver-Initiated MAC Protocols for Energy Efficient WSNs. In *Proceedings of the 18th Nordic Conference on IT Systems (NordSec)*, pages 1–16. Springer, 2013

[26] : Xenofon Fafoutis, Dušan Vučković, Alessio Di Mauro, Nicola Dragoni, and Jan Madsen. Poster Abstract: Energy-Harvesting Wireless Sensor Networks. In *Proceedings of the 9th European Conference on Wireless Sensor Networks (EWSN)*, pages 84–85. University of Trento, 2012

[27] : Xenofon Fafoutis, Alessio Di Mauro, and Nicola Dragoni. Sustainable Medium Access Control: Implementation and Evaluation of ODMAC. In *Proceedings of the 4th Workshop on Energy Efficiency in Wireless Networks and Wireless Networks for Energy Efficiency (E2Nets)*, pages 407–412. IEEE, 2013

[28] : Xenofon Fafoutis, Alessio Di Mauro, and Nicola Dragoni. Sustainable Performance in Energy Harvesting Wireless Sensor Setworks. In *Proceedings of the 4th International Conference on Future Energy Systems (EENERGY)*, pages 267–268. ACM, 2013

[29] : Xenofon Fafoutis, Alessio Di Mauro, Madava Dilshan Vithanage, and Nicola Dragoni. Receiver-Iinitiated Medium Access Control Protocols for Wireless Sensor Networks. *The International Journal of Computer and Telecommunications Networking*, 76:55–74, 2015

«Always pass on what you have learned»

Jedi Master Yoda

Acknowledgments

At last, the acknowledgments. This part is the last thing I wrote in this dissertation, however for most of you it will be the first part you read. Heck, for many of you it will probably be the *only* part you will ever read. So, I better make a good impression and say some clever things. The truth is that, if you know me, this is not my strongest suit. I have already struggled (but hopefully managed) to keep serious for the one hundred-odd pages to follow, so I guess the only way for me to keep some dignity would be not to write anything at all and to encourage you to skip this altogether. However, as you have probably already figured, I will not do that. The actual dissertation is the result of three years of work, sweat, blood and a broken bone. This, on the other hand, is where I get to say silly things and make unreasonable claims without having to corroborate them with scientific citations, scatter plots or convoluted algorithms. I mean, considering that D. J. Bernstein has compiled a twenty-one slides demonstration involving rings, fields and Minkowski's theorem to prove that the number 83 is prime (true story, look it up), and that Proposition $*54.43$ of *Principia Mathematica* is an eight-line quasirandom arrangement of symbols which proves that $1 + 1 = 2$, I should at the very least have a couple of chapters proving how come that when you decide to get up from the table at the canteen, fifteen more people decide to do exactly the same thing at precisely the same moment, and we all end up at the disposal area, entangled to form a chaotic contraption, trying not to throw forks into the food-waste bin. I know it would probably make for an interesting read on social interaction, human

psychology and to some extent fluid dynamics, but I think I will spare you with all that, and actually cut to the chase.

There have been many people who stood next to me in this adventure and it would be hard and way too long to list them all, without considering that I would probably forget somebody and feel really bad about it. For this reason I have decided not to name names, but rather to address everybody as a collective (see how many things research has taught me? I get to cheat even in my acknowledgments page). For all the fun moments, the laughter, the late beers. For the crazy nights and the cooking sessions. For all the advice, the help and the good words. For keeping up with my personality and my quirks. For making my life better and for being there for me when I needed. For the chicken, the tea and the potatoes. For your communication addiction disorder. To the old ones that are still here, and to the new ones that had the misfortune of finding me on their path, I hope it was not so bad. For staying around and being the most unbelievable bunch of crazy people that I had the honor to meet. For all this and much more to come, this one is for you. I humbly thank you, my Friends!

Alessio

La dernière chose qu'on trouve en faisant un ouvrage est de savoir celle qu'il faut mettre la premiére. (The last thing one knows when writing a book is what one should put in first.)

Blaise Pascal

1 Introduction

Over the last years and thanks to the improvements of technology, computers and embedded devices have become more and more ubiquitous. It is nowadays normal and almost expected for everything that we use in our everyday life to have some kind of intelligence. From our computers, to our phones and tablets, everything that surrounds us is interactive and very often interconnected.

The miniaturization process has drastically increased the possibility of fitting such smart devices into more and more items. Smart houses, smart cars and smart systems in general make our lives easier and leave us always in control of the swarm of devices that we own or that we find scattered throughout the places that we visit every day. We are often able to control all such devices at once, independent of what our physical location is, thanks to them being all networked and remotely accessible.

Wireless sensor networks (WSNs) are a collection of small embedded systems

that can communicate one another to perform many different tasks. They perfectly fit in this description of ubiquitous systems and are becoming more and more widespread. They are deployed to solve many problems and their applications are disparate. Used to effortlessly monitor extensive areas in an unmanned fashion, finding their ways in military applications, or granting remote safe access to hazardous locations, WSNs are a formidable way to address many challenges.

Scientific research on this topic has been going on almost since the dawn of computers. It is however with the transition to the digital world we now live in, that we can really start to take full advantage from this technology. It is now possible to make our abode more comfortable by making sure that the habitat that we will find therein will be precisely matching our requirements, even if we have been absent for a long period of time. Or again, our car will constantly monitor itself thanks to the large number of sensors disseminated throughout the whole vehicle, and aptly inform us of any malfunction, all while we are comfortably sitting behind the wheel.

Historically speaking, power has always been a major concern for WSNs which had to be serviced and kept operational after the initial deployment, something that usually required human intervention. Thanks to the all-new introduction of energy-harvesting wireless sensor networks (EH-WSNs), this factor has been greatly mitigated by allowing individual nodes to recharge themselves through solar panels, wind turbines or similar technologies. While far away from being a magic wand that solves all the problems, the recent addition of energy harvesting (EH) has dramatically improved the lifespan of a sensor network, making modern devices far more reliable. New applications have also been made possible, up to the extent where miniaturized nodes have be implanted into the body of a person to monitor the sugar level of his or her blood, while exploiting the oxidation of the very same sugars to power themselves.

Thanks to EH, the general paradigm of *achieving the longest possible lifespan while still producing a good enough result* has shifted in favor of *producing the best possible result with the amount of energy currently available*. The fact that energy will become available once again in the near future allows a node to undertake energy intensive

tasks and tap into its reserve, knowing that it will soon be replenished.

Finally, given the increase in usability, number of functionalities, and the stricter and stricter dependability requirements, security for WSNs is a major concern. Given the degree of trust that we put into these systems, making sure that the information that a node carries should not be made available to external unauthorized entities, or that these same information and parameters can be modified only by those that have the right authority to do so, is paramount. These and many others are the goals of computer security, something that should be factored in the design of every good application, right from the beginning.

1.1 Dissertation Overview

Given the incredible impact that the addition of EH has had on the applications relying on WSNs, we set out to understand whether or not the same kind of impact can be had on the security aspect. Is it possible to take advantage of the EH capabilities of a node to provide better, more reliable and more efficient security? This and other similar questions constitute the main focus of this dissertation, which tries to give these problems an answer.

The reminder of this manuscript is organized as depicted in Figure 1.1 and as discussed here. In Chapter 2 we give an introduction to WSNs, explain in greater details how they work by analyzing each one of their components individually. We will explore what different types of WSN do exist, what are their main characteristics, advantages and disadvantages. We will also talk about the main challenges and design goals of sensor networks. Additionally, in this chapter we will introduce EH-WSNs, take a closer look at how they solve many problems posed by regular WSNs but also how they introduce a whole new set of challenges.

Chapter 3 follows naturally from the previous one. Here we present an introduction to security in the context of WSNs, we discuss what we mean by the term *secure system* and talk about the need of a comprehensive classification of attackers. To this purpose we introduce a new and extended taxonomy of attackers that takes into account the specific challenges of EH-WSNs. We then apply our taxonomy

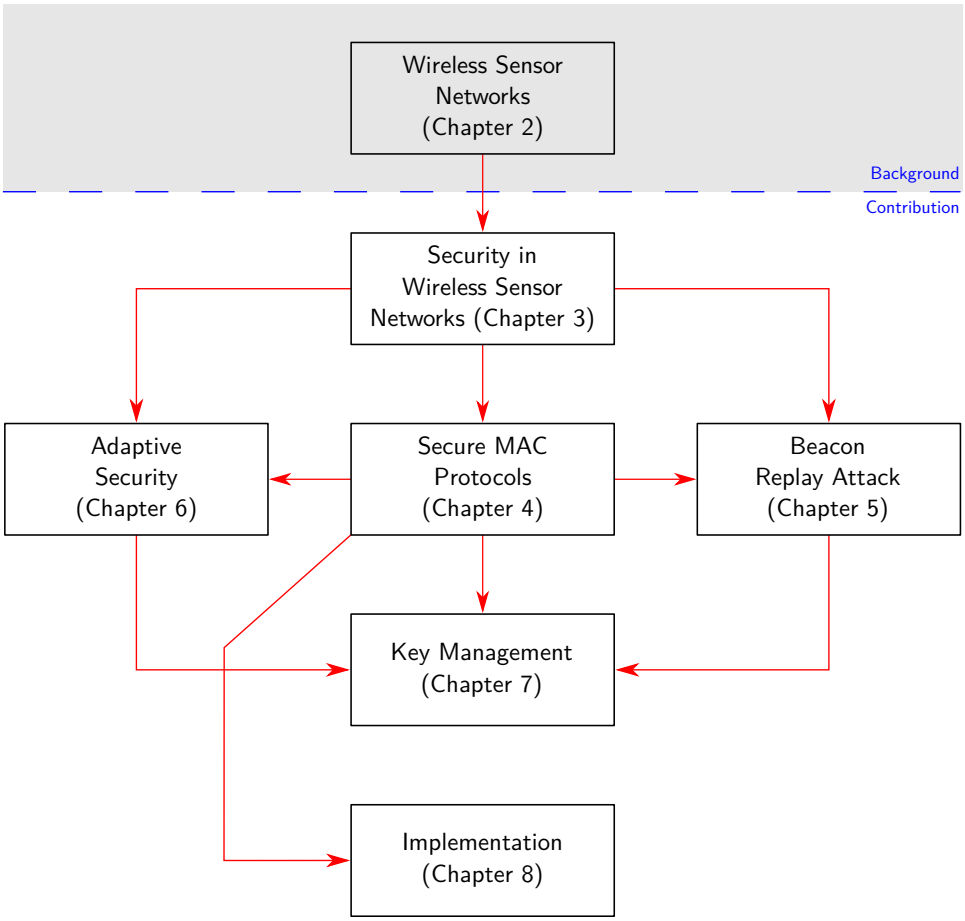


Figure 1.1: Organization and logical connections.

to standard well-known security attacks that can be mounted in a sensor network, and compile a list of attackers describing the capabilities required to perform them. We then introduce one of the main topics of our work which is security in the data link layer. We describe it at a deeper level by introducing different types of medium access control (MAC) protocols and then presenting its state of the art and some previous work.

Chapter 4 focuses even more on the data link layer, and MAC protocols. Here we present on-demand medium access control (ODMAC), a new receiver-initiated (RI) protocol specifically designed for EH-WSN. We then discuss its security and

how it can be improved. To that extent we introduce a security suite designed for and incorporated into ODMAC. We finish this chapter by introducing an all-new attack, the beacon replay attack.

Chapter 5 continues right where the previous chapter leaves. Here we in formally describe and introduce the beacon replay attack first introduced in Chapter 3, a new attack specific to ODMAC and to the whole class of RI protocols in general. We discuss why this attack is significant, what can be achieved from it and why classic solution for similar attacks do not work. We then present the receiver authentication protocol (RAP), an adaptive protocol that can be used to defeat the beacon replay attack and, at the same time, be configured in many different ways, to suit both regular and EH sensor networks. We finally present a security proof of our protocol and analyze its performance in terms of energy consumption.

In Chapter 6 we shift our attention to adaptive security and take advantage of the concept introduced in Chapters 3 and 4. Here we discuss how the constantly changing energy levels of an EH-WSN can be leveraged from a security standpoint. We look at how different schemes and parameters can be used according to the current energy status of the whole network but also on a link-to-link level. We discuss some different modes and configurations that can be used to achieve different results and provide a discussion about how the scheme can be freely adjusted to match the constraints posed by the applications and how we believe this should be the main driving factor in general. We finish the chapter by discussing the soundness of the scheme.

In Chapter 7 we introduce the topic of key management. This complements the discussion points presented in Chapters 3 to 5 as a fundamental component for securely using encryption. Here we present an introduction to some canonical approaches for distributing and managing cryptographic keys in a sensor network. We then present a specific scheme by the name of multipath key reinforcement in greater detail. We discuss how it can provide better security but how it is not a good match for EH-WSNs. We then present a new version of this scheme that addresses the problem and takes full advantage of the different energy levels of an EH-WSN by means of an adaptive approach. We conclude the chapter by describing some

experimental work and the related results.

Chapter 8 contains our practical implementation work. In this chapter we describe the hardware platform used and talk about the capabilities of each component. We then present and discuss the implementation of ODMAC and how this has been realized. We continue by presenting and justifying the main experiments that we have been running on our platform, and discuss some analytical results.

Finally Chapter 9 concludes the dissertation providing an overview of our findings, discussing some future work and presenting some final remarks.

1.2 Main Contribution

The following list contains the main contribution presented in this dissertation.

Attack Taxonomy and Cyber Physical Attacker We present an attack taxonomy to take into account new aspects of security introduced by EH. We apply this taxonomy to common attacks and argue for the need of a new attacker model that is specific to sensor network.

Security suite for ODMAC We discuss and introduce ODMAC and expose its lack of security. We then design and introduce a specifically tailored security suite that can provide authentication and confidentiality.

RAP We define the beacon replay attack, a new attack relevant to the entire class of RI MAC protocols. We then introduce a new protocol that defeats this attack. We test the protocol both formally and analytically.

Adaptive security scheme for MAC protocols and EH-WSNs We present the foundations for a new scheme that allows to achieve different security levels and security properties within the network, depending on the amount of energy currently available.

Key management We introduce a new extended key reinforcement scheme to be used with EH-WSNs. We follow two approaches, one static and one fully dynamic and present the connected results.

Implementation We present some implementation work connected to ODMAC and its security suite.

It's that simple. Wireless is wireless, and it's digital. Hopefully somewhere along the line somebody will add more ones to the zeros. When digital first started, I swear I could hear the gap between the ones and the zeros.

Eddie Van Halen

2

Wireless Sensor Networks

The main focus of this work are wireless sensor networks (WSNs) so it is appropriate to begin the dissertation by introducing what a WSN is. We will present the most relevant aspects, but for more detail and additional information the reader is referred to [1, 87].

2.1 Introduction to WSNs

WSNs are collections of small and inexpensive embedded devices, normally referred to as *sensor nodes* or just *nodes*. These nodes are interconnected through a computer *network* which relies on radio communications, thus making it a *wireless* network. Typically nodes are scattered throughout a geographic area that is known as the *sensed area*. Such an area can span from the body of a person up to entire fields and forests.

In its simplest incarnation (Figure 2.1) a WSN has two main types of components: several sensor nodes and one special node called *sink* or base station (BS). Let us analyze them both.

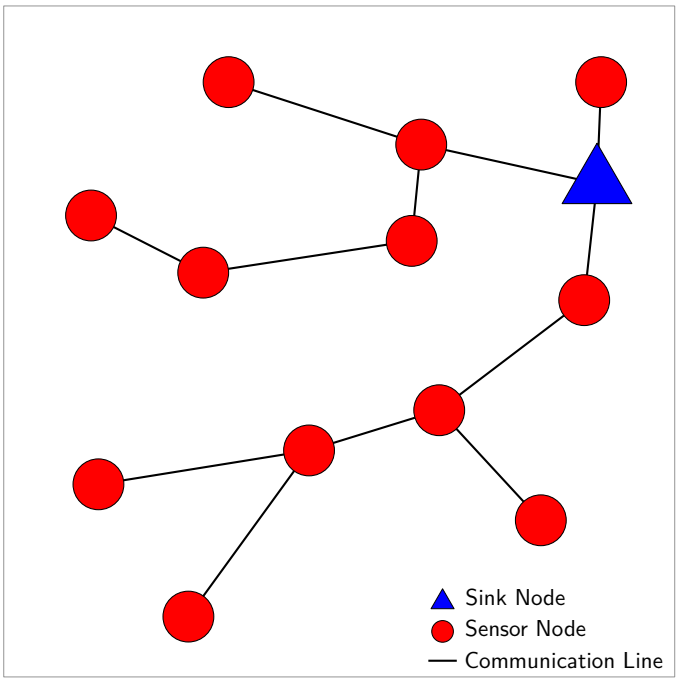


Figure 2.1: A typical wireless sensor network.

2.1.1 Anatomy of a Node

Nodes are the core building blocks of WSNs and are equipped with a few fundamental components as shown in Figure 2.2.

First of all a power supply unit (PSU) is required to enable the whole apparatus to work, this is typically a battery of some form and the related regulating circuitry, but as we will see later on, notable exceptions to this concept do exist.

A second crucial component is a processing unit, which provides the nodes with some kind of computational power. Typically microcontroller units (MCUs) are used for this purpose, but microprocessor units (MPUs) can be found as well. Independent of the actual technology in use, this component is the brain of the sys-

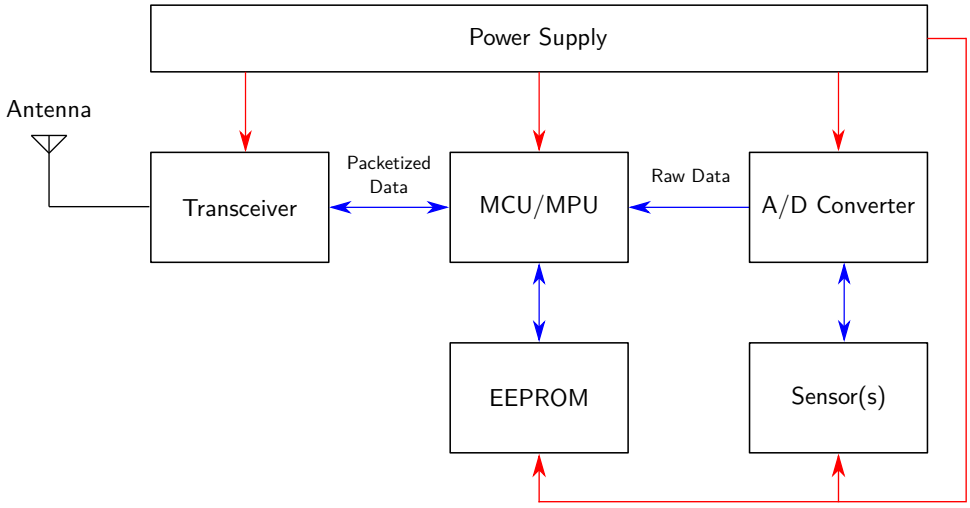


Figure 2.2: The main components of a sensor node.

tem and gives nodes the capability of being programmed and thus perform non-trivial tasks on the data at their disposal, rather than myopically relaying information.

A third important component is memory, this is needed for each node to store programs and data. Depending on the specific processing unit used, this component might be stand-alone or built-in.

In order to fulfill their role as data generators, nodes must be equipped with one or more sensors. These are almost always integrated circuits (ICs) that allow to transform a physical quantity such as temperature, humidity, radioactivity or light intensity, to name a few, into an electric voltage or current that in turns can be converted by means of an analog to digital converter (ADC) into a digital value, understandable by the processor and ready to be elaborated or transmitted. Again, depending on the specific technology used for the sensors, the ADC might be a separated unit or it might be built inside the IC package.

Besides generating data, nodes also have a second role, that is acting as a network relay. In order to fulfill this a radio frequency (RF) transceiver is necessary. This component, with the help of an external antenna, provides a node with wireless connectivity allowing it to communicate with other nodes by using predefined

schemes or protocols.

A typical modern and mass produced sensor node can be seen in Figure 2.3.

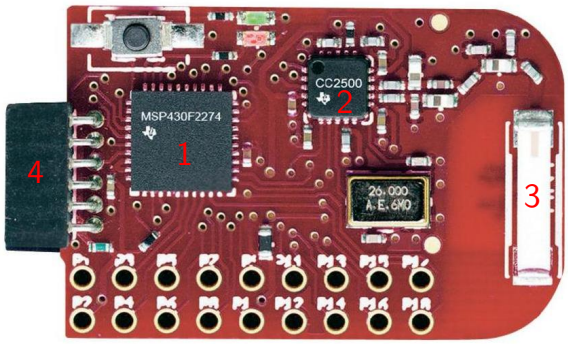


Figure 2.3: A typical, mass produced, sensor node (ez430 [40]) with its main components: the MCU containing the integrated memory, the ADC and a temperature sensor (1), the transceiver (2), the on-board antenna (3) and the power supply connector (4).

Sensor networks can survive and operate correctly only if enough nodes are active, in order to do so duty-cycles (DCs) must be introduced. Given that the amount of energy available to each node is a fixed an constantly decreasing quantity, it is highly inefficient to keep every node of the network switched on without any reason. Similarly, performing too many operations such as reading from the sensor too often, can also quickly deplete the battery. The technique used to extend the life of a node is known as *duty-cycling*. DC is a term borrowed from electronics engineering and defines the percentage of a period where a periodic signal is active.

Assuming that nodes perform tasks in a cyclic fashion, in the world of WSNs the DC is defined as the amount of time that the node spends being active over the length of its period. When a node is not in its active state, it is effectively turned off or *asleep*. Modern MCUs are equipped with several low-power modes (LPMs) modes where different peripherals are powered down. By selecting one of these LPMs modes, the node can maintain active just its essential components, keeping ready to be awakened when a specific event is fired. Duty-cycling is a standard but fundamental technique that is used in virtually every sensor network.

2.1.2 Sink Nodes and Base Stations

The other important entity that can be found within a WSN is the so called sink. This is a special node that greatly differs from regular sensor nodes, both from a physical and a logical point of view. The sink node, also referred to as BS is a much more powerful and capable component of the system, usually a laptop class entity. Hence it can perform much more complex tasks for extended periods of time.

Normally the role of a sink is to be the final recipient for all the data packets generated within the network, collecting them and functioning as an interface toward the end users which can be either human or other hosts on a different kind of network. Anyway this is a general scheme and may vary according to the specific network topology and organization.

2.1.3 Design Goals

There are several design goals in WSNs and they encompass many different aspects. First and foremost a sensor node should be inexpensive. It is not uncommon for a network to have hundreds of nodes which are considered disposable assets. Very rarely individual nodes are serviced or even replaced after a failure. This would not be possible if a single unit was expensive to produce.

Secondly a node should be small in size. The reason for this are multiple: smaller nodes are cheaper to mass manufacture and usually have a lower power consumption. Sometimes this is also dictated by the application itself. If a sensor node has to be injected into the bloodstream of a patient it must necessarily be small enough not to obstruct the veins.

Another very critical quality is low energy consumption. As we will discuss later, energy plays a big role in WSNs. Nevertheless, it is intuitive to understand that the lower the energy consumption, the longer a whole network can survive.

More important qualities are adaptability and self-configurability. Sensor networks are highly dynamic systems where the neighborhood of a node is constantly changing due to the fact that single nodes might be able to physically move throughout the network or that they are expected to fail over time. This means that a robust

design which allows complete independence is key. Nodes should react and adapt when new peers join or part the network, they should be able to self-configure, forming the predefined topology in a completely distributed manner, without having to rely on external help during deployment phase.

Reliability and fault tolerance are other important qualities. RF links are susceptible to errors and interference, while nodes could suddenly fail. Well designed sensor networks should be resilient to this phenomena ensuring a correct information exchange by means of correction codes and retransmissions, and ensuring the survivability of the service despite unexpected node failures.

Finally, security is another big concern. Depending on the specific application the data exchanged among the nodes might be sensitive. Furthermore, sensors are often deployed in harsh environment and make use of standardized wireless schemes and encodings. This allows potential attackers to gain unfair advantage, acquiring information that should be kept secret or preventing a service from working correctly.

2.1.4 Constraints and Challenges

WSNs can be used for a wide array of applications, however their particular nature poses notable challenges during the design phase of protocols and networks in general. As it often happens, the typical challenges of a system clash, with the design goals defined before.

The most important constraint of a sensor network is energy and it constitutes the driving factor for many design choices. As we discussed before, in a typical WSN nodes are powered by batteries. This means that energy is a scarce and precious resource that is inevitably going to deplete over time. For this reason optimizing how the energy is spent is a fundamental problem for WSNs. Optimization can be achieved in many different ways, from increasing the power density and the efficiency of the cells within a battery, to dynamically adjusting operating voltages and frequencies for the individual chips, to designing energy-aware protocols that gracefully scale depending on the amount of residual energy.

Another constraint that is characteristic of WSNs is computational power. Nodes are equipped with highly optimized controller units, specifically designed to run on low power embedded devices. A typical node uses processors running at frequencies of 8-16 MHz and have on-board memory that goes from a few kilobytes to a few megabytes. The trade-off is that usually the performance must be somewhat sacrificed to achieve acceptable energy consumption. As a result, the average computational capabilities of a sensor node are limited and so is the complexity of the tasks they can carry out. This ultimately impacts on the quality of software designed for WSNs.

A third source of limitation is the transceiver itself. A transceiver is a complex unit containing several non-trivial components which impact on the performance and the price of a device. Furthermore, RF communication is a sophisticated technology with significant path loss, susceptible to interference and very energy demanding. This causes the transmission range of a sensor node to be between a few meters and a few hundred meters.

Last but not least one of the driving factors of WSNs is cost. Sensor nodes are meant to be small, inexpensive and easily replaceable. This further constraints the decisions that can be taken when designing, implementing and deploying a WSN.

2.2 Energy-Harvesting WSNs

As discussed before, energy is one of the main constraint of WSN. Most of the decisions taken when designing a sensor network are guided by the constant struggle for energy, trying to put to good use every single unit of power. Thanks to the advancement in technology and research, a new game-changing design has been introduced: energy-harvesting wireless sensor networks (EH-WSNs).

EH-WSNs are regular WSNs where the main battery is supported by additional energy sources, or even entirely replaced by them. The key idea is to reclaim the energy present in the surroundings of a node, convert it to a usable form through appropriate types of transducers and ultimately employ it to power the node itself.

If the energy source used to fuel the network is abundant and always present

then, from a theoretical point of view, energy harvesting (EH) provides a network with unlimited energy and an almost infinite lifespan, where the considerably less common hardware failures become the main concern. As it is easy to imagine this ideal scenario does not always match reality. Despite being a significant improvement over regular sensor networks, EH-WSNs have their own unique challenges.

A comparison between the energy usage pattern of regular and EH nodes is shown in Figure 2.4.

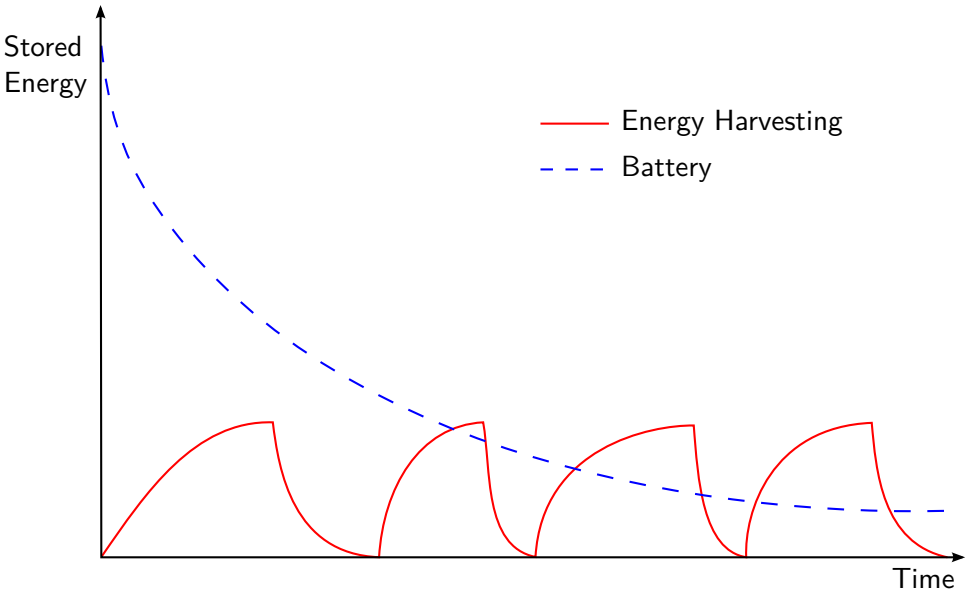


Figure 2.4: Energy usage comparison between battery operated and EH sensor notes. The energy within the battery monotonically decreases, whereas the EH node replenishes itself over time and uses the energy once there is enough of it available. [73]

2.2.1 Types of Harvestable Energy

The Law of Conservation of Energy states that «*The total amount of energy in an isolated system remains constant over time*». Energy however has many different forms, and conversion techniques are never 100% efficient. This has two main contrasting repercussions. First of all, if we were able to design a machinery that was able

of perfect efficiency, i.e., transferring all the energy at the input to the output, we could not have EH technologies because there would simply not be any energy to scavenge in the first place. On the other hand if perfect efficiency is not an option it means that some of the energy must be dissipated into other forms during the conversion process. As a result we will have to harvest more than the nominal value of energy required by the node to run in order to keep it powered.

If we were to measure the efficiency of different energy conversion processes and we assume that the best known harvesting technology is always used, it is possible to rate how laborious it is to obtain a given amount of energy depending on the source used. In other words, not all energy forms are created equal and some of them are inherently harder or less convenient to harvest.

By far the most efficient and easy to harvest form of energy is solar, which relies on to the photovoltaic effect. Using modern solar cells it is possible to harvest tens of milliwatts per centimeter squared. However, this figure is only valid for near-optimal situations where the solar cell is under direct sunlight. In indoor scenario with artificial lighting the average value drops by several orders of magnitude. Solar technology is the most predominantly used form of EH and is present in many applications.

Another form of energy that can be harvested is wind. The airflow can be used to activate small rotors or turbines which will then output power. The amount of power achievable with this technology is around one milliwatt per cubic centimeter.

Electromagnetic radiation can also be harvested through special antennas called rectennas and can provide a few hundred microwatts.

An interesting form of energy to harvest is vibrations. This is related to a mass vibrating at a specific frequency for a given amount of time. Despite the basic idea being the same, there are three main ways to harvest vibrations. The first is by using the piezoelectric effect and piezoelectric materials such as quartz or Rochelle salt. This kind of materials are able to accumulate electrical charge in response to mechanical stress. For example by inserting a piezoelectric transducer inside a shoe the kinetic energy of a footfall will deform the transducer which in turns will pro-

duce energy that can be harvested. A second way of obtaining energy through vibrations is by electrostatic conversion. This technique consists of externally charging the two plates of a capacitor and subsequently varying their distance. This will result in a variation of charge that can be used to produce electrical energy. This technique can be used to harvest energy from industrial machinery that vibrate while operating. The final method used in conjunction with vibrations is to produce energy through electromagnetism. By moving a magnet inside a coil, an electrical current is generated. This last technique can be used for example to harvest environmental vibrations. Depending on the technique used, the amount of energy attainable through vibrations harvesting ranges from a few microwatts to some milliwatts in highly specialized applications.

Another source of energy commonly used is the one obtainable from thermal gradients which rely on the Peltier-Seebeck effect: when two dissimilar metals with different temperatures are joined together, a voltage is generated. The amount of energy that these sources can deliver varies from several tens to a few hundreds of microwatts per unit size.

Finally, it is possible to harvest more exotic forms of energy such as the oxidation of blood sugar or the metabolic energy of trees. However these are far less common are are only used in very specialized applications.

Energy sources can also be described according to their characteristics [76] and how easy they are to obtain. It would be highly impractical to have an extremely good harvesting architecture for a source of energy that originates from highly infrequent phenomena. We define an energy source to be *controllable* if it can be obtained when it is required, whereas *non-controllable* forms of energy can be scavenged when some external conditions are met and the specific event manifests itself. Similarly we define an energy source to be *predictable* if it manifests itself with some kind of periodicity or if it possible to know its presence in advance. *Unpredictable* energy sources behave in the exact opposite way and manifest themselves without any pattern.

This concepts can be applied to the same form of energy arising from different sources. While it is always true that solar energy is a non-controllable but pre-

dictable form of energy, vibrations can have different configurations. Imagine the vibrations produced by people moving inside an airport. In this highly complex environment the energy source is non-controllable and unpredictable. On the other hand consider the vibrations given off by some kind of machinery (e.g., a rock tumbler), here presence of the energy directly relates to when the machinery is turned on, making it a controllable source. Furthermore, the machine could follow a predetermined operating cycle yielding a predictable source. Table 2.1 contains a condensed view of some examples of the different energy sources.

Energy Source	Author	Normalized Power (μW)	Volume (mm^3)
Vibration - piezoelectric	Glynne-Jones et al. [33]	0.5	125
Vibration - piezoelectric	Roundy et al. [71, 72]	28	1,000
Vibration - piezoelectric	Roundy et al. [72]	50	1,000
Vibration - piezoelectric	Marzencki et al. [56]	0.0007	2
Vibration - electrostatic	Mitcheson et al. [58]	0.005	750
Vibration - electrostatic	Despesse et al. [16]	27	1,800
Vibration - electrostatic	Despesse et al. [16]	1.7	32
Vibration - electromagnetic	Shearwood et al.[74]	4.7×10^{-8}	5.4
Vibration - electromagnetic	Glynne-Jones et al. [34]	7.7	840
Vibration - electromagnetic	Perpetuum Ltd. [53]	25,000	30,000
Thermoelectric	Applied digital solutions [10]	50	41
Airflow	Park and Chou [64]	50,000	100,000
Solar (outdoor)		20,000	500,000
Solar (indoor)		1,500	500,000

Table 2.1: Comparison of different sources of energy. [32] [76]

2.2.2 Energy Usage

Another peculiar aspect of EH-WSNs is how the energy is actually consumed. This can be done according to the architecture, which can be either *harvest-use* or *harvest-store-use*. As it is possible to see in Figures 2.5 and 2.6 the latter requires additional components to be realized.

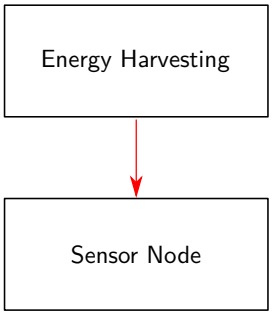


Figure 2.5: The harvest-use paradigm. The energy coming from the harvesting unit is directly used to power the node.

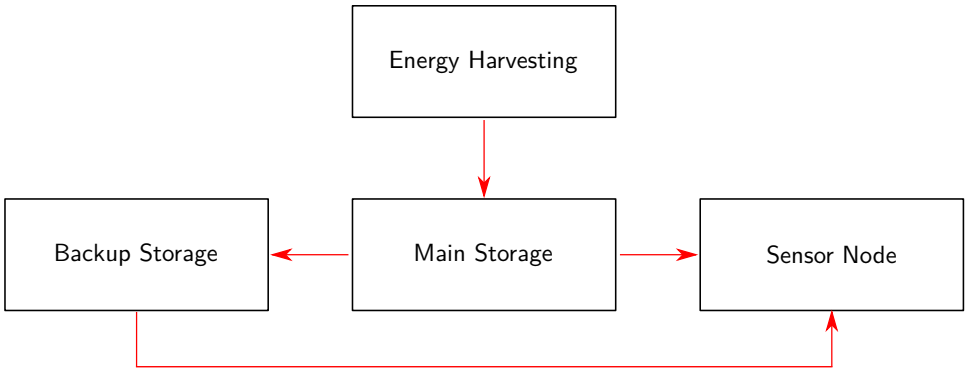


Figure 2.6: The harvest-store-use paradigm. The energy coming from the harvester is stockpiled into the main storage unit. Optionally a backup unit can be used for long term storage.

In the harvest-use architecture all the energy scavenged is directly employed to power the sensor node. As a result, as soon as the energy level falls below the minimum operating threshold, the node will shutdown and come back online only when enough energy is available again. Another caveat of this architecture is that all the energy that is not used by the system is wasted. This means that in case of abundant energy it is not possible to stockpile any of it.

On the contrary the harvest-store-use architecture addresses this limitation by introducing an energy storage device. This device acts as a buffer in a producer-consumer paradigm: it is charged by the energy harvester (producer) and discharged by the node (consumer), effectively decoupling them. When the produc-

tion rate is less than or equal to the consumption rate, then the harvest-store-use architecture behaves like harvest-use. On the other hand if the energy available is more than the energy required, then the excess can be saved to be used at a later time. Additionally, a second backup storage device such as a rechargeable battery can be used for even longer term storage.

These two architecture have different characteristics and they are suited for different scenarios. Imagine an application based upon non-controllable and unpredictable energy source such as monitoring traffic by having cars run over a piezoelectric strip on the tarmac. In this case the average energy level will be low and nodes will only be energized by the presence of a car hitting the transducer. This application is well suited for an harvest-use architecture and does not benefit from the increased cost of having additional components. If we instead think about a temperature monitoring application, out in a field and powered by solar energy, here we can see how having an energy buffer can help keeping the network alive and functional even during nighttime. The downside of this approach is the increased cost and complexity of a single node.

2.2.3 Energy Neutrality

As discussed before, the energy consumption of a node is connected to its activity, which depends upon many factors. The most effective way of adjusting it is by varying the DC. Keeping a node switched off obviously saves energy, however it also means that less tasks can be carried out. The balance between tasks performance and energy consumption is known as energy neutrality [82]. More specifically if at a given harvesting rate and activity level, a node can remain powered on for an indefinite amount of time, then it is said to be in energy neutral operation (ENO) state. If besides achieving ENO a node can also achieve maximum task performance, the node is said to be in the maximum performance energy neutral operation (ENO-Max) state.

As a result, in the context of EH-WSN, tracking the consumption of a node and varying its DC is a good technique for achieving the ENO state. By defining and

minimizing an appropriate cost function, it is possible to obtain an optimal duty-cycle schedule. For example the cost function suggested in [82] is

$$\lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} (B_t - B_0)^2$$

where $B_0 \in [0, 1]$ is the initial battery level and B_t is the battery level at the discrete time step t.

The ENO state is a key property of EH-WSNs, and every protocol and implementation should strive to achieve it.

2.3 WSN Classification

A WSN can be structured and organized in one of several different ways, according to the characteristics and the number of its component. We will now describe some of the most common configurations and arrangements for WSNs.

2.3.1 Network Topology and Architecture

As showed previously in Figure 2.1 the typical architecture consists of a single sink, connected to the sensor nodes via a *single-hop* or *multi-hop* network. The difference is that in single-hop networks (Figure 2.7) each node can directly communicate with the base station. This can greatly simplify the protocols required for the network to function. However, the size of a single-hop network is also limited by the communication range of the nodes. On the contrary, in multi-hop networks the distance between a node and the sink can be greater than one hop, hence the name. Multi-hop networks can cover much greater areas, but they require message forwarding infrastructures and non trivial routing algorithms making the whole system more complicated and, in general, more energy demanding.

Multi-hop networks can be further subdivided according to their internal organization. In a network where all the nodes are equal and have the same duties we talk about a *flat* network, this has the same arrangement shown previously in

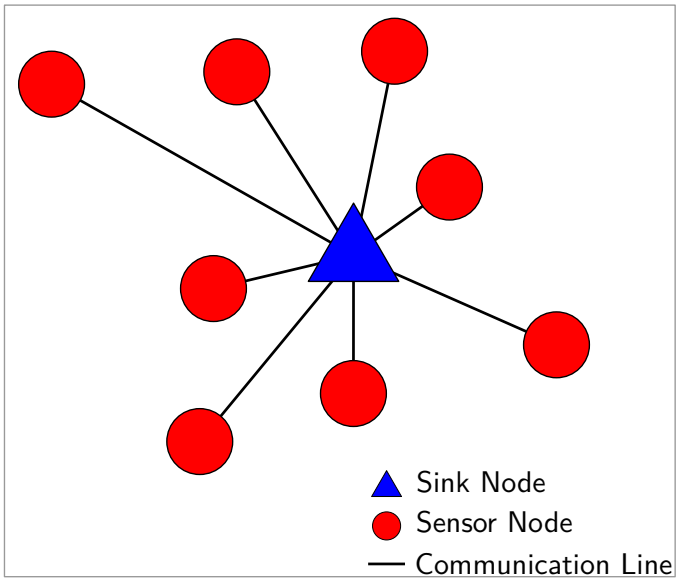


Figure 2.7: Single-hop architecture, each sensor node is in direct reach of the sink node.

Figure 2.1.

On the other hand in a *hierarchical* network we have a subdivision into one-hop groups called *clusters* where regular nodes relay their messages to special nodes called *cluster heads* which are responsible for the whole group (Figure 2.8). Cluster heads will then form a higher tier network to communicate and forward messages to the sink. The difference between regular nodes and cluster heads can be both logical and physical. For example we could have a dynamic protocol that elects cluster leaders in a distributed manner, e.g., according to their energy level. The process is then repeated for a cluster when the head is running out of power. Another possibility is to use a network with special nodes that have increased energy capabilities (i.e., more batteries or larger batteries). These nodes will be used as cluster heads. The added complexity of a hierarchical network is payed off by advantages in terms of performance and scalability. Each regular node has to communicate (generally) with a one-hop cluster head, and will not have to forward any message from other nodes. Data aggregation is also typical in this kind of architecture. The cluster head can consolidate multiple messages into a single one, either

by means of an aggregation function such as averaging, or by grouping messages into a single super-message and transmitting that in one operation. Additionally, load balancing is also an option, increasing the scalability of large networks.

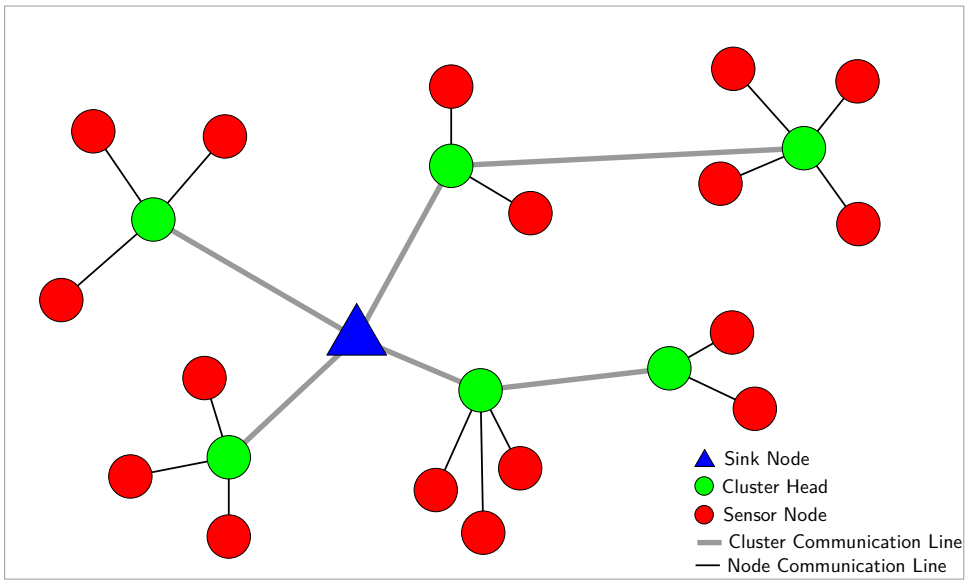


Figure 2.8: Multi-hop hierarchical architecture. The cluster heads form a higher tier layer.

It is worth noting that the concept of clustering can be further expanded in two directions. In one case it is possible to have a multi-hop architecture within a single cluster, while the second possibility is to have multi-tiered architecture by subdividing clusters into lower lever clusters up to an arbitrary number of times.

2.3.2 Types of WSNs

Regardless of the specific architecture, the overall organization of the network is also depending on the specific characteristics of its component.

Starting with mobility, individual nodes may or may not have the ability of moving around the network. In one case we talk of *static* networks whereas in the other case we have *mobile* networks. In static network the connection graph formed by the different nodes is well defined and does not change over time, except for nodes

permanently leaving the network due to hardware failures or battery depletion. This allows to compute routing tables once and use the for most of the lifespan of the network. However, static networks might become disconnected if a large enough number of nodes, or few hub nodes, cease to work. This is most notable for the nodes in close proximity of the sink, which will have to handle a considerably higher amount of traffic compered to peripheral nodes. On the contrary, in mobile networks, the nodes can move around either in a predictable manner, for example in the case of motorized autonomous nodes or in an unpredictable fashion, such as nodes attached to animals or people. This creates an additional layer of complexity and increases the cost, especially if the nodes are not moving "for free" on an independent autonomous carrier, but require dedicated hardware. Depending on the application, the routing could be simplified and effectively reduced to a single-hop network if all the nodes can consistently get in direct transmission range of a collection point (i.e., sink).

The concept of mobility can be applied to BSs as well. As with static networks, in the case of *static-sink* networks the connection graph is well defined and similar techniques can be used. In *mobile-sink* networks is the BS to move around in order to collect data. The advantage of this technique is that the added complexity of the system is on the sink node and not on every single sensor node. With this configuration the nodes do not require any modification and can wait for the sink to get in range and dump all their collected data. One disadvantage of this technique is that if the collection rate is not high enough nodes might run out of memory to collect data. Depending on the application, this problem can be mitigated by aggregating the data.

The most common kind of sensor networks only have one sink and is therefore called *single-sink*. This is an inexpensive solution and makes data collection easier for the end user. However, depending on the size of the network, it causes a considerable amount of packets to be relayed by other nodes, resulting in significant energy consumption. By adding other sinks we have what is called a *multi-sink* network. These are effectively separate networks where each node reports to its closest BS. If the different sinks are positioned in strategic points the load of the

network can be better balanced and its lifespan increased. Furthermore, having multiple sinks also provide some redundancy allowing the networks to survive a sink failure by recomputing its routing paths. A disadvantage of this technique is the increased cost, especially considering that a single BS is orders of magnitude more expensive than a sensor node.

Another way of classifying networks is according to the placement of the nodes. If nodes are deployed in random positions within the sensing area, we talk of an *unstructured* network. This is the norm for WSNs as it is generally unfeasible or too expensive to carefully plan the positioning of each node. However, if this is done, we have what is known as a *structured* network. In this case it is generally possible to take advantage of the placement of the nodes and optimize the network in some way, for example having the largest area coverage with the minimal number of nodes.

Finally, a network can be either *non-adaptive* or *adaptive*. In the former case nodes are statically configured and are not able to self-organize into a network. Non-adaptive networks are normally used only in small deployments where the number of nodes is contained. On the other hand, adaptive networks do not require any external intervention, and can organize into a fully functioning network by themselves. This is a very useful property since, as discussed, WSNs are highly dynamic, and nodes can part and join at a considerable rate.

As it can be seen there are many ways of configuring a sensor network. However, in the end is always the application that dictates the constraints, what can and can not be done or what kind of configuration is the best match for the scenario at hand. This is a key fact in WSNs that, as we will see in the rest of this dissertation, will show up time and again.

2.4 Protocol Stack

The internal organization of a single node, and by extension of the whole network, is fundamental to understand how a WSN works and how it can achieve better performance.

2.4.1 Layer Organization

Similarly to regular computer networks, WSNs tend to be organized in layers. Like in the open systems interconnection (OSI) stack, each layer makes use of the services from the layer underneath and provides services to the layer above. The typical layers of a WSN are: *Physical, Data Link, Network, Transport* and *Application* as shown in Figure 2.9.

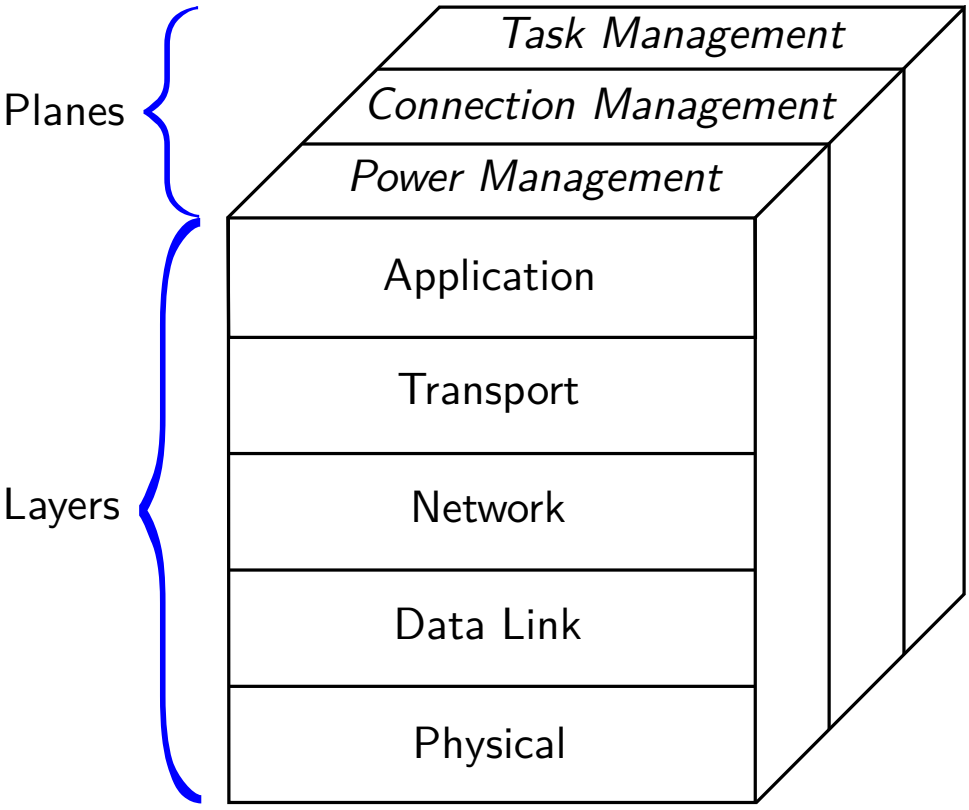


Figure 2.9: Layers and planes organization of a single node. [1]

The physical layer is fundamentally the interface between the node/sink and the outside world. The goal of this layer is to convert back and forth between bits and radio signals. Among other things this layer will have to concern itself with choosing an appropriate transmission band and frequency, producing the modulation and demodulation and implementing techniques such as clear channel as-

sessment (CCA) and received signal strength indicator (RSSI). These can be used to analyze the energy level of a given spectrum and compare it to its noise floor to establish whether or not the medium is busy with an ongoing communication. Furthermore, RSSI can also be used to estimate the distance of a node.

Data Link layer has as its main tasks to establish connections (generally pairwise) between nodes in direct communication ranges. This layer is fundamental for ensuring a fair usage of the medium while addressing concerns such as energy consumption, communication delay and throughput. How and when the medium is accessed, how contentions are resolved and what to do in case of collisions is all decided in the subsection of the data link layer known as medium access control (MAC).

Moving up we find the network layer. This layer is responsible for routing, that is establishing connection between any two nodes that need to communicate and that are separated by two or more hops. It is here that routing tables are built and maintained by taking into account numerous parameters coming from the specific application such as network topology, usage patterns, network density and so on.

Transport is the next layer and it is responsible for providing reliability to the connection. Depending on the application different properties could be required. A few examples are delivery guarantee, in-order delivery and congestion control.

The final layer is application. This includes the main application running on each node. Currently there are no standardized application protocol and as a result each specific application can be considered as a different implementation of this layer.

2.4.2 Plane Organization

Together with layers, in Figure 2.9, we find an orthogonal division into planes. This has become more and more common over time. Planes represent portion of the system dedicated to addressing specific tasks and achieving specific goals, independent of the layer division.

The most common planes are *Power Management, Connection Management* and

Task Management.

The power management plane is responsible for administering the energy reservoir of a node. For example it can suggest when to turn off the radio in order to save power, or which neighbor to choose in order to obtain an energy efficient route.

The connection management plane is used to allow single nodes to organize into a network. It can dictate how the nodes should self-configure upon joining and how previously existing nodes should react. Similarly when nodes part the network due to them being mobile or because of some kind of failure, this plane suggests how the reconfiguration should happen.

Finally, we have the task management plane which is used to optimally share the different task of the network among all the nodes. This is done by taking into account their capabilities and the parameters of the network as a whole including the topology and how the actual phenomena under observation are organized.

2.4.3 Holistic Approach

Although the layer division used in WSNs is similar to the one used in regular networks, it is highly inefficient to use the same protocols unmodified. The reason for this is that the protocols used in regular networks have fundamentally different goals. More complex functionality such as automatic retransmissions and quality of service (QoS) are provided. However, the price to pay for this is an increased energy consumption, which, as we discussed, is a premium resource in WSNs. To address the problem, many specific protocols for each different layer have been developed and are used in different applications. However, in some cases the layered approach used in the highly specialized protocols is more of a legacy than a conscious choice. In this direction, the introduction of planes is a tentative to move away from an airtight compartmentalized approach and towards a more *cross-platform* solution.

Taking this a step further leads to what is usually referred to as *holistic* approach. Here the layer division is much more relaxed and limited to the core functionality of each layer, whereas any other relevant information is made available to every

other subsystem so that interaction and optimization can be maximized. Since WSNs are extremely energy sensitive, being able to incorporate this information in as many places as possible and having it guide as many decisions as possible allows for better performance across the whole system.

Regardless of what approach is used, there are chances of running into security issues. This is what we will discuss in the next chapter.

The explosion of companies deploying wireless networks insecurely is creating vulnerabilities, as they think it's limited to the office - then they have Johnny Hacker in the parking lot with an 802.11 *antenna using the network to send threatening emails to the president!*

Kevin Mitnick

3

Security in Wireless Sensor Networks

Many times WSNs and EH-WSNs are used to monitor and convey sensitive data or to run dependable services. It is not uncommon for security to be critical within a WSN application and therefore, over the past years, it has been a hot research topic.

Just like in regular computer security the three most important properties and goals of a good security suite are *confidentiality, integrity* and *availability*. These are well known security concepts and their meaning is the following

Confidentiality access to data and resources should be granted only to the rightful actors

Integrity the data present within the system should never be inappropriately modified, neither intentionally nor because of unpredictable errors

Availability services and information should be available whenever they are re-

quested

There are many other properties that can be achieved by a secure system, however the so called *CIA* properties are the most widely accepted "minimum requirements". These definitions are used because they capture the essence of the problem (e.g., not to have eavesdroppers on a confidential channel), but are also general enough to abstract from the implementation and the specific details.

Security is a highly complex topic which touches upon many aspects of a single system, from the software to the hardware, trough each one of the layers. Being able to describe required properties with one definition, independent of how the actual property is realized, is fundamental.

3.1 Attacks and Attackers for (EH-)WSNs

We start our analysis by discussing the importance and need for a well organized and formally usable classifications of attacks and attackers.

3.1.1 The Importance of Classification

Security in the field of WSNs is an extensive topic and can be tackled from a wide variety of angles and through many different aspects. A useful tool to have in this case is a thorough classification, a taxonomy of attacks more specifically. Thanks to that it is possible to systematically describe the key properties that an attacker needs in order to mount specific attacks, analyze how the composition of such properties allows for the implementation of more complex attacks, and identify where to intervene in order to have maximum efficiency. Furthermore, a classification is required to set the path for formal methods, security analysis and automatic provers.

This approach is not new, however, the addition of EH capabilities gives a new spin to the topic, requiring new considerations, introduction of new properties and adjustments to old ones.

3.1.2 A Taxonomy of Attacks

The taxonomy that we developed [19] is specifically tailored to include EH-WSNs. The key idea is to divide possible attacks using so called *dimensions*. A dimension defines a parameter and helps with analyzing what an attacker can do by varying each parameter. The three main dimensions are *time*, *presence* and *intervention*.

The time dimension defines the duration of an attack, this spans from a few seconds for eavesdropping a message or jamming the channel, all the way to many hours or days in the case of offline brute-force attacks. Presence describes the position that an attacker can occupy within a network and, as described also in [55], ranges from *local* where only one or a few components of the network can be affected, to *distributed* where multiple local components are interested, all the way to *global* where the attacker has complete simultaneous access to the whole network. The third dimension, intervention, defines the possible actions that an attacker can take against a system. Extending the ones defined in [5], we introduce the following possibilities:

- **Disabling** - the ability to temporary remove nodes from the network;
- **Destruction** - the ability to permanently disable one or more nodes;
- **Eavesdropping** - the ability to listen to and store exchanged messages;
- **Data Knowledge** - the ability to acquire data from one or more nodes;
- **Partial Data Modification** - the ability to interfere with the data stored within a node (usually through external access);
- **Complete Data Modification** - the ability to completely and arbitrarily change the data content of a node;
- **Reprogramming** - the ability to completely and arbitrarily change the code content of a node;
- **Node Injection** - the ability to add nodes, under the control of the attacker, to the network;

- **Energy Reduction** - the ability to starve a node, preventing it to use its main energy source;
- **Energy Exploitation** - the ability to maliciously take advantage of the current energy level of a node;

Unlike what presented is in [5], the new list of interventions does not form a lattice under the *requires* ordering. Instead, as displayed in Figure 3.1, it is a lower semilattice given that there is not a supremum, mainly due to the addition of the energy related interventions.

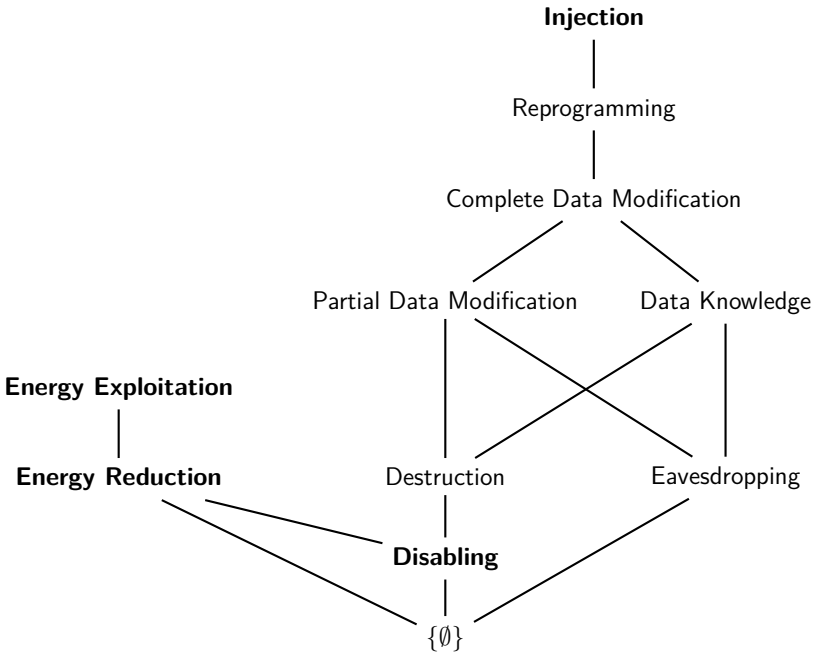


Figure 3.1: Lower semilattice of the interventions, extension of [5].

By mixing and matching different properties and capabilities it is possible to generate various attacker models. As said before this can be done systematically and procedurally in order to obtain provable properties about a given system. Take for example the famous Dolev-Yao attacker model [21], here the adversary can eavesdrop, capture and forge any message. This model can be represented as a

combination of *{(Global, Eavesdropping), (Local, Reprogramming)}*, using the first part to model the fact that its knowledge of the exchanged messages is total, and the second to model the fact that it can be an insider. A preliminary result of this classification is that EH-WSNs and the attacks possible therein, can not be fully modeled by the Dolev-Yao attacker. Let us briefly. discuss why

3.1.3 The Cyber-Physical Attacker

The commonly adopted Dolev-Yao attacker model is a good starting point for regular and EH WSNs, however this model is sometimes too powerful for this setting while it does not cover other important aspects. Threats like data knowledge and reprogramming must be taken into account. If we analyze a network purely from a protocol definition point of view, it is possible to see that if an attacker is capable of Eavesdropping, Data Knowledge, and some kind of Data Modification, he can effectively reprogram a node, but with less effort than having to have physical access.

Being able to know how a legitimate node would react to a specific type of messages gives the attacker a non negligible advantage towards breaking the protocol, for example by forging appropriate malicious responses. Moreover, due to the unstable nature of EH-WSNs, destruction capabilities may not be needed. By taking advantage of the fact that nodes might be in a temporary sleeping state due to lack of energy, it is possible to mount what we later on introduce as the *sleepwalker attack*, where a node is maliciously replaced by the attacker only during the period of time while it is inactive. At the same time, the idea of energy reduction does not produce Destruction like in regular WSNs, but rather it yields only Disabling because of the harvesting mechanism.

Intercepting messages might also require some adjustments. Given the broadcast nature of sensor networks it is hard for the attacker to reliably receive a message while preventing any other node from doing so. If a node is physically close enough to the attacker, it will receive the same messages that the attacker will receive and preventing that would cause both of them not to receive anything.

The notion of globality must also be applied in a slightly different way since, again, it is possible for large portions of the network to be effectively inactive for very long periods of time, or simply the sheer size of a sensor network might make this objective unfeasible. For example monitoring the traffic of the entire network might prove to be considerably costly.

Generally, as a result we believe that by combining the properties listed in our taxonomy is it possible to define a *Cyber-physical attacker*, that is indeed required in order to obtain correct formal models and to analyze security properties. This is out of the scope of this dissertation and is further discussed in [18] and more extensively in [81].

3.2 Taxonomy Application

We now take advantage of the taxonomy we introduced in the previous section to classify and describe some of the classic attacks in WSNs [42, 84] and their respective attackers. We will follow a layer-oriented approach using the definitions introduced in Section 2.4.1. The idea here is to characterize and formalize these well-known attacks by breaking them down into the interaction of basic components.

Attacks will be described as a set of tuples where each component identifies the dimensions of time, presence and intervention. For simple attacks only one tuple is used, while in case of more complex attacks each element of the set will model a specific aspect.

Further, a general overview and classification of the presented attacks is displayed in Figure 3.4.

3.2.1 Physical Layer

We start our classification at the bottom of the stack with the physical layer. This layer allows to establish the physical link between two nodes and intervenes in the selection of appropriate frequencies, modulation schemes and symbol generation among the others.

We start with *eavesdropping* which can be considered an attack in and of itself. Thanks to the broadcast nature of RF, an attacker with a receiving radio tuned to the same frequency used by a transmitting node can listen on and record any ongoing communication. If the traffic is not encrypted and the attacker knows the protocol it is possible for him to make sense out of if. Even if the protocol is unknown but the traffic is still unencrypted, it is possible to reverse engineer the packet format. The adversary required for this attack is very simple and is defined as {(*, *, Eavesdropping)}. This means that the more time and effort the attacker puts in, the more effective the attack will be.

Similar to the previous attack we have what is normally called *traffic analysis*. The way of performing this attack is the same, a receiver is used to listen to the channel to obtain information. However, the main goal in different. While in eavesdropping the attacker set out to gather data from each packet, with traffic analysis the goal is to understand how the traffic is shaped, in which direction do the packets travel, which are the nodes that handle the highest number of packets, whether there are nodes that exclusively communicate with a specific subset of other nodes, etc. This can be performed by counting the messages exchanged and by looking at the meta-data usually contained within the header of a message. The attacker required to mount traffic analysis is the same described before {(*, *, Eavesdropping)}.

The two attacks described so far are normally referred to as *passive attacks*. This is because these attacks do not cause direct harm to the system and the adversary is merely an observer. However, they provide precious information about where more harmful *active attacks* should be directed. By knowing which are the high value targets it is possible for the attacker to better organize and focus his efforts. All the attacks that we will describe from here on are active attacks and therefore will have practical repercussions on the network.

The third attack for the physical layer is *jamming*. As we pointed out before, the wireless medium is broadcast by nature and given its specifications anyone could intentionally interfere with the communication between two nodes simply using a strong enough transmitter that can reduce the signal-to-noise ratio below accept-

able levels, thus preventing data exchange. Being this a physical attack, it can be classified in many different ways according to the hardware used by the attacker. The time dimension can span all the way from short to long and the presence can go from local to, theoretically, global. Only the disabling intervention is required. The attacker can then be modeled as {(*, *, Disabling)}.

Another attack related to the physical layer is *tampering*, which consists in gaining access to the content of a node with the possibility of modifying such content. The tuple for this attacker is {(*, Local, Reprogramming)}. On the other hand for a milder version of tampering when the attacker can only dump the content of a node {(*, Local, Data Knowledge)} will suffice.

3.2.2 Data Link Layer

The data link layer is appointed to establish a link between two nodes within direct communication range. It is a fundamental layer for WSNs and we will discuss more about it in the following sections.

The first attack that can be mounted in this layer is *collision exploitation*. Knowing the underlying protocol it is possible for an attacker to send the right message at the right moment, causing a collision and making packets unintelligible. This might trigger costly re-transmission procedures, decreasing the overall throughput or effectively removing a node from the network. Assuming that the protocol knowledge is an offline effort for the attacker, the resulting tuple is {(Short, Local, Eavesdropping), (Short, Local, Disabling)}.

A direct consequence of the previous attack is called *exhaustion*. Technically the attack implementation can be considered the same, but the goal is to deplete the energy of a node through repeated collisions and therefore the associated attacker is {(Short, Local, Energy Reduction)}.

Another important attack is the so called *replay attack*. Here overheard data packets are sent unaltered either to the same target node or to a different one, in order to exploit non idempotent operations such as path construction in routing algorithms and data aggregation in applications. The attacker required to perform

replay should be {(Short, Local, Eavesdropping), (Short, Distributed, Partial Data Modification)}.

We will discuss the replay attack in further details in Chapter 5, where we will introduce a new version of it that we call the *beacon replay attack*.

3.2.3 Networking Layer

The next layer is the networking layer whose purpose is to connect two nodes that are not directly in range and hence require the forwarding of packets through multiple links or hops.

In order for routing algorithms to work properly, appropriate information must be propagated throughout the whole network. This is a typical entry point for *spoofing* attacks where the values are somehow forged or altered, so that imprecise information are sent out. This attack can have several effects: use of sub-optimal paths, introduction of routing loops and, in some cases, even the partition of the network. For an attacker to deploy such attacks it is required to feed the victim nodes with fake information. This yields the tuple {(Medium, Local, Partial Data Modification)}. The time dimension is dependent of the actual protocol used and how the information are propagated.

Next we will describe a series of attacks that build upon the possibility of modifying routing information. They all share a great deal of similarity, but each one of them has slightly different goals.

The first one is known as the *sinkhole* attack, which allows one single node to attract all or most of the traffic within the network. The attacker might not be able to access the content of the collected packets, but this is not a key requirement.

If besides collecting messages the attacker also drops them all, the resulting attack is the so called *blackhole* attack. This kind of attack is however very invasive making it easier to discover. Nonetheless, if the attacker has enough knowledge about the network and wants to selectively take out specific nodes it could deploy a *selective forwarding* attack, where only the messages belonging to specific nodes are dropped.

More involved attacks can be performed, for example the *sybil* attack in Figure 3.2 is achieved by advertising multiple identities to other nodes. Depending on the networking algorithm used, this could have a major impact on routing tables and path calculation.

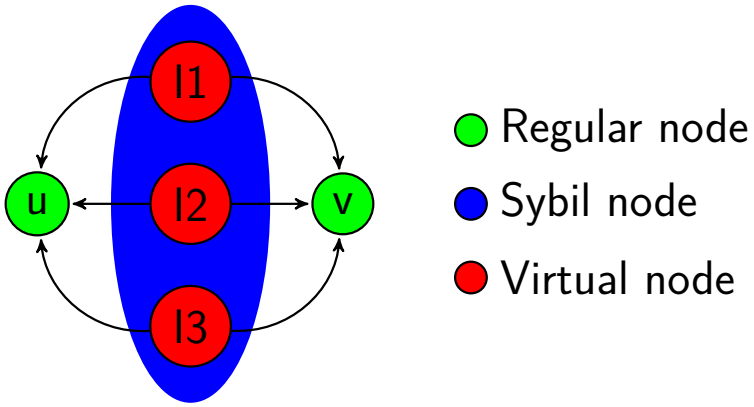


Figure 3.2: Sybil attack, a Sybil node (blue) sends messages to regular nodes (u, v) claiming different identities (I1, I2, I3).

Another quite effective attack is the so called *wormhole* attack (Figure 3.3) where two or more nodes collude to make packets appear in different portions of the network at a much faster speed than what would normally happen by simply relaying those packets through other nodes. This is typically obtained by using an additional low latency channel between the attackers.

The first result of this is that some nodes are mislead into believing that there are specific nodes in their immediate neighborhood when this is not the case. Again, depending on the networking algorithm employed, these nodes could represent better choices from a routing point of view and the victims of the attack may try to communicate with them directly. The result in this case are dropped packets due to *time-to-live* expiration or similar mechanisms taking place.

The sinkhole attack only requires knowledge about what packets to capture, therefore its related attacker requires {(Long, Local, Reprogramming), (*, Global, Data Knowledge)}. The other attacks instead aim at disrupting other nodes more actively and they substitute Data Knowledge for Partial Data Modification.

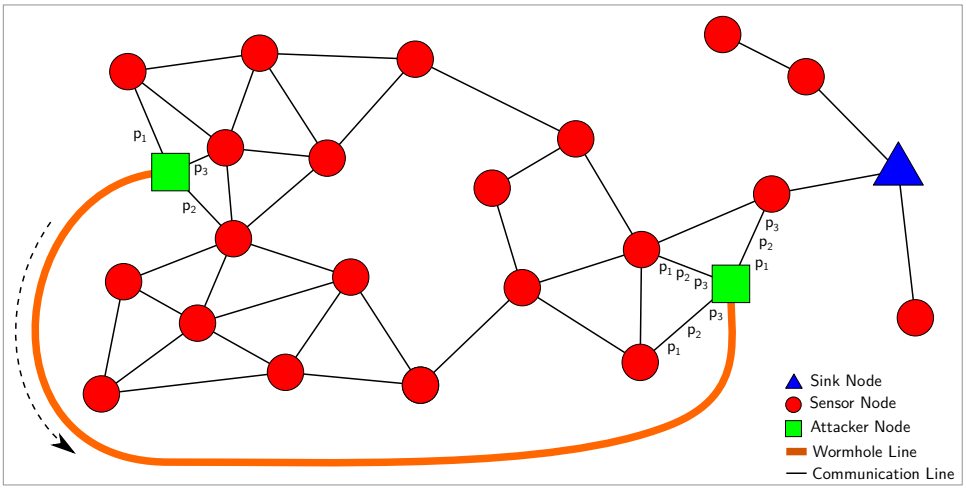


Figure 3.3: Wormhole attack, here two nodes collude through a low latency channel to quickly transfer packages across the network and invalidate routing information.

All these attacks can be mounted by the same general attacker who has the capabilities discussed above: {(Long, Local, Reprogramming), (*, Global, Partial Data Modification)}.

3.2.4 Denial of Service

A special mention must be made for denial of service (DoS) attacks. These are attacks against the availability of the service which use many different techniques. Because of the fact that each layer and component of a given service must be working in order for the whole service to be available, DoS can be implemented in many different ways at each different layer. For example an attacker with physical access to a node could render it unusable and thus preventing any kind of application from running.

Another possibility is to exploit the MAC protocol, this can be done by selectively jamming specific control messages that allow nodes to synchronize, or just by advertising an extremely long transmission thus making all the other nodes to power down for a considerable amount of time.

The blackhole and wormhole attacks that we introduced before are other forms of DoS attacks. By dropping all the packets, either directly or by taking advantage of the expiration mechanism of the underlying routing protocol has the direct effect of shutting down all the communications and therefore the whole application is compromised.

As a result, each different method and technique used for achieving DoS can be modeled in a slightly different way. However, if we abstract from the actual implementation of the attack and focus on its main goal of undermining availability, we can derive two types of attacker. In one case we have a physical attacker that can access enough nodes and permanently disable them, this is defined as {(Medium, Distributed, Destruction)}. On the other hand we have a non-physical attacker that has enough knowledge of the protocols used by the application, that he can take advantage of them by preventing the exchange of messages for a sustained period of time. This kind of attacker is defined as {(*, Distributed, Disabling)}.

3.2.5 Attacks Specific to EH-WSNs

While these attacks are general for regular WSNs, new ones can be described specifically for EH-WSNs. They do not fall in the description of typical attacks, but we would like to introduce the concept here for the sake of completeness.

For example the approach used in [79] chooses opportune encryption algorithms and key lengths according to the amount of energy currently available. Furthermore, some QoS is provided by assigning priorities to different packets and using the available energy to relay high priority ones first. If we introduce here an attacker with {(Medium, Local, Energy Exploitation), (Short, Local, Eavesdropping)} it would be possible to passively analyze the traffic according to the current level of energy and monitor what is the typical behavior of the system when nodes have low energy, assuming that the messages exchanged in this state will be almost only high priority ones. This might also reveal a fruitful strategy in case that some of the nodes are involved in a high number of transmissions of important message, as these might be worth attacking further. Moreover, if we allow the attacker also

the possibility of energy reduction, it would be possible to maliciously lower the amount of energy of specific nodes, in order to force a particular mode which may include weaker encryption algorithm or a shorter key.

Another example is the idea presented in [65], where the authors note that by using stream ciphers it is possible to precompute the key stream. This allows for a trade-off between memory and energy. Some bytes can be precomputed when the energy availability is high, and used up when it is low. Depending on the communication scheme used in the network an attacker defined by the tuple {(Short, Local, Data Knowledge)} could obtain the precomputed bytes and use them in the future. The attack can be made even more effective if we also allow energy reduction with {(Medium, Local, Energy Reduction), (Short, Local, Data Knowledge)}. By controlling the energy amount of a node the attacker is forcing its victim to use the precomputed stream effectively behaving like {(Medium | Long, Distributed, Data Knowledge)}, but with considerably less effort.

Finally similar attacks can be re-implemented and require different attackers in the EH scenario. For example the exhaustion attack defined before requires a continuous effort from the attacker and goes from {(Short, Local, Energy Reduction)} to {(Long, Local, Energy Reduction)}.

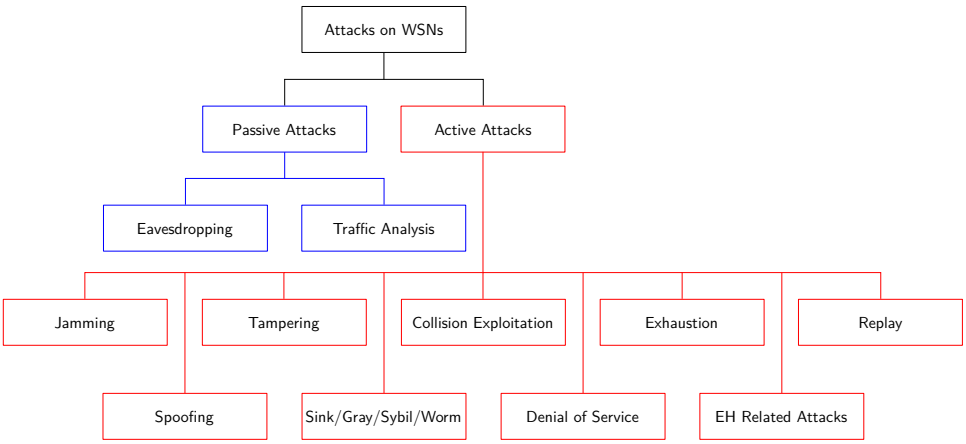


Figure 3.4: Overview of the presented attacks.

3.3 Security of the Data Link Layer

After having discussed known and common attacks in general, we will now delve into one of the layers of WSNs, namely the data link layer and analyze it in greater detail. MAC protocols belong to this layer and are delegated to establishing a connection between two neighboring nodes, i.e., two nodes that are physically in range one another. As a result MAC protocols have the all important task of controlling the radio activity: deciding when to send and receive messages, handling collisions and taking care of re-transmissions. These tasks are central for the energy efficiency of a sensor network, in fact the radio is by far the most energy demanding component [3], as can be seen in Figure 3.5 hence a proper handling can guarantee considerable energy savings. Furthermore, the problem of maintaining the radio switched on while awaiting for incoming traffic, commonly known as *idle listening*, is also addressed by MAC protocols.

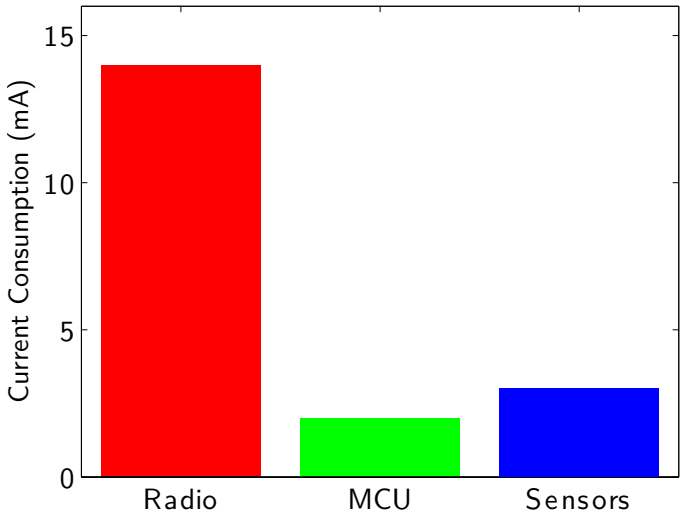


Figure 3.5: Typical energy consumption of the components of a sensor node during activity (Radio CC2500 and MCU MSP430) [3].

As a result, security in MAC protocols is also very important. Being able to manipulate control messages enables an attacker to mount a wide variety of attacks as described in Section 3.1.

3.4 Data Link Layer and MAC Protocols

In order to understand and design effective security solutions for MAC protocols, it is important to have some insight on their key properties and how they work in general. We will discuss this next. Therefore, we enter into a short digression to explain how MAC protocols work.

The very first MAC protocols used to keep the radio component constantly on. While this was a near-optimal scenario in terms of throughput and delay, it yielded extremely high energy consumption. As a result this behavior quickly ceased in favor of a technique that allowed nodes to enter low-power or sleeping modes according to specific parameters. We introduced this before as *duty-cycling* and it provides significant energy savings. However, while being extremely effective, duty-cycling, makes establishing a connection much harder. The problem of finding a neighbor to connect to is now augmented in the time direction since such neighbor might be inactive when a message is ready to be sent. Sophisticated techniques have been developed to address this issue.

Typically, MAC protocols that perform duty-cycling are organized in two large categories: *synchronous* and *asynchronous*, depending how duty-cycling is performed. In the former, nodes wake-up and fall asleep with a predetermined frequency while in the latter there is no such constraint and the nodes are free to self-organize. Synchronous protocols usually provide good communication guarantees, but at the cost of higher energy usage because nodes are forced to wake up and perform a run of the protocol even it no message is present. Asynchronous protocols on the other hand are much more flexible from that point of view and allow pairs of node to establish a specific duty-cycling pattern that can optimize message exchanges.

Moreover, asynchronous protocols are further divided into two subcategories known as *preamble based* and *beacon based* or, more commonly, *sender-initiated (SI)* and *receiver-initiated (RI)* respectively. The three different approaches are summarized in Figure 3.6.

In the preamble based scheme, whenever a node wants to transmit a frame it will first send a message called *preamble*, and then wait for a potential receiver to wake

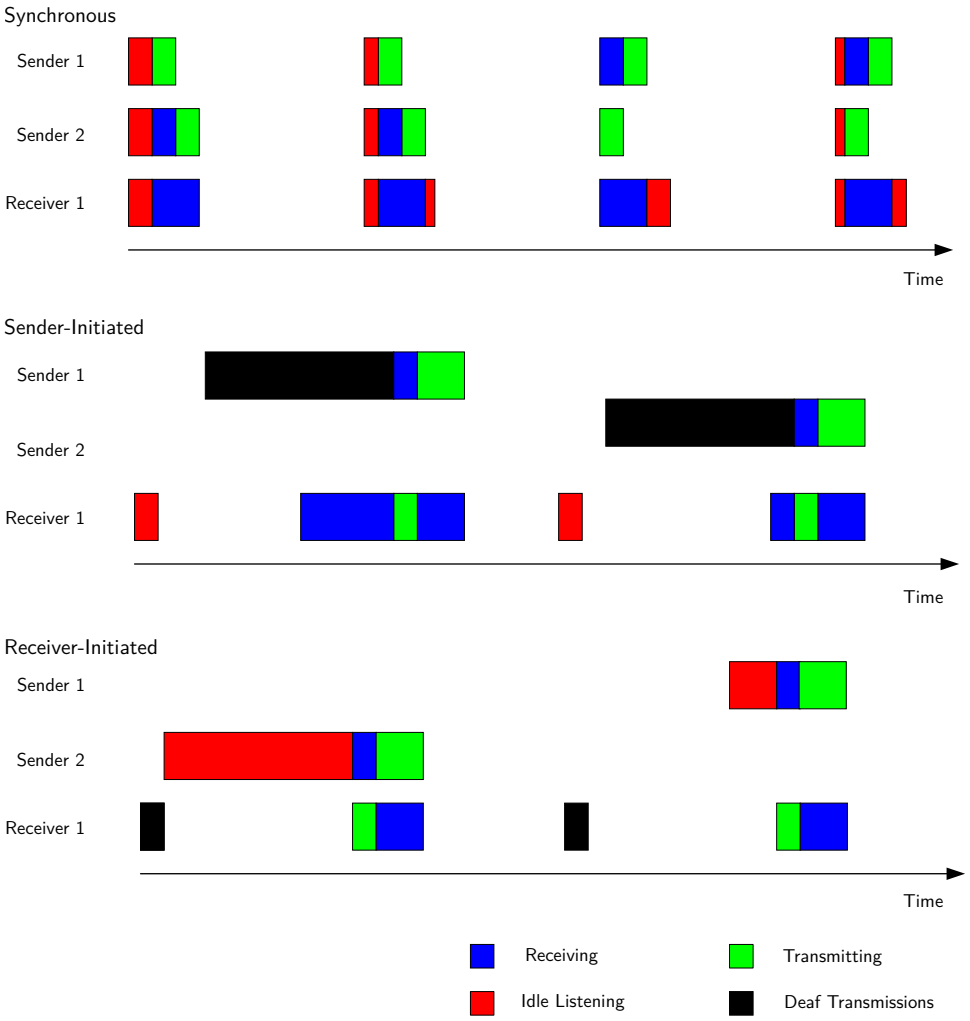


Figure 3.6: Three different approaches to using duty-cycles: synchronous, sender-initiated and receiver-initiated.

up. When and if this happens, the receiver answers by means of an acknowledgment (ACK), thus interrupting the preamble and establishing a connection. The main problem of this technique is that the listening for an ACK and transmitting the message afterwards can take a considerable amount of time, and require for the on-board transceiver, the most energy demanding component, to be switched on for the entire period.

On the other hand we have the RI paradigm, first introduced by Lin et al. [51]

in 2004 and later popularized by Sun et al. [77] in 2008. This family of protocols acts in a somehow specular manner when compared to SI. Instead of sending out a long preamble, a node willing to *receive* data will issue a very short message, called a *beacon*. This message will inform potential senders of the presence of an available receiver. As shown in [25, 51, 77], this leads to a more efficient energy usage and, in turns, lower consumption.

3.5 Security Suites

As it is normal to expect, the MAC layer is also the building block for every networking protocol since a path between any two nodes is nothing more than a collection of links between neighboring nodes. As a result of this, security at the data link layer is extremely important and a number of solutions have been developed over the years to address that. The most well-known security suite for WSNs are SPINS [67] and TinySec [43]. Others also exist and we will discuss them briefly.

3.5.1 SPINS

SPINS is a complete security suite for sensor networks that provides confidentiality, authentication, integrity and freshness. We discussed the first three properties in previous sections. With regard to data freshers it is informally defined as a guarantee that the messages sent within the network are recent and are not duplicate of previous messages. SPINS is composed by secure network encryption protocol (SNEP) and micro timed efficient stream loss-tolerant authentication (μTESLA).

The first component, SNEP, is a low overhead encryption protocol. It provides semantic security (also known as indistinguishability under chosen plain-text attack (IND-CPA)), data authentication, replay protection and weak message freshness. SNEP makes use of a single symmetric cipher as building block for all the cryptographic operation such as encryption and authentication. The cipher used in the original implementation is RC5 [68] and cipher block chaining message authentication code (CBC-MAC) is used for message authentication. Separate keys are used for encryption, authentication and random number generation. Each one

of these keys is derived from a master key shared by the BS and the node.

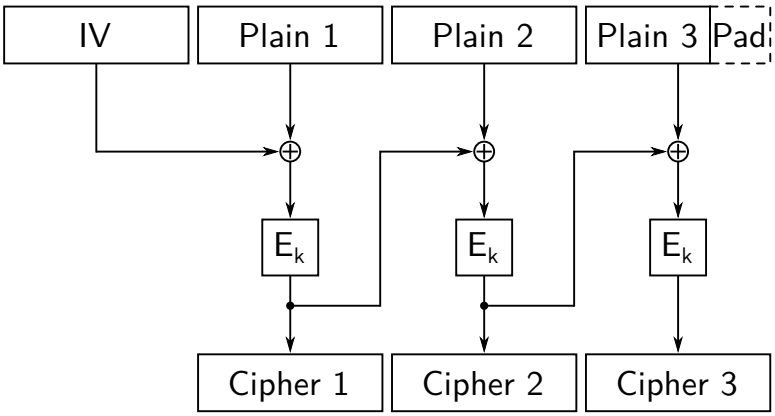


Figure 3.7: Encryption with CBC mode of operation. Each block is *xor*ed with the previous block before encryption. If the last block is not a multiple of the block-size it may cause cipher-text expansion.

Each pair of nodes exchanging information maintains a counter that is increased by one after each successful message exchange. The counter is implicit, in the sense that it is never transmitted, but in case of desynchronization, a separate exchange protocol is run. This is how SNEP achieves freshness. Furthermore, the counter is also used as part of the input to the encryption process thus providing semantic security.

If required, strong freshness (a version of freshness that provides a total order on a request-response pair) can also be obtained. This is done by having the sender generate and transmit the authenticated version of a random nonce and a request message. The receiver answers with the authenticated version of the same nonce and a response message. Upon verifying the authentication code, the sender knows that the response has been generated after the request was sent.

The second component, μTESLA, provides authenticated broadcast and is a more energy efficient implementation of TESLA [66], with shorter key-chains, capped number of senders and use of symmetric encryption in order to fall within the constraints posed by WSNs. μTESLA is based on *hash chains* and requires a loose time synchronization among the nodes. In order to send an authenticated

packet, the sender generates the authentication code at the current interval using a secret key. The message is sent and in a following interval the key is disclosed. At this point a receiver can use a previous key to authenticate the new key, and in case of a correct match use the new key to authenticate the message. No confidentiality is implemented so the protocol does not encrypt the messages.

3.5.2 TinySec

TinySec [43] is another energy efficient security protocol, specifically designed for the data link layer of WSNs. It is integrated into the default implementation of TinyOS [49], an operating system for WSNs. It provides data authentication and message confidentiality. TinySec does not include protection against replay attacks.

Similarly to SNEP, also TinySec uses a single symmetric cipher to perform all the cryptographic operation. The algorithm used in this case is Skipjack [63] in cipher block chaining (CBC) mode, while CBC-MAC is used for authentication. In order to avoid cipher-text expansion, a technique called cipher-text stealing (Figure 3.8) is used.

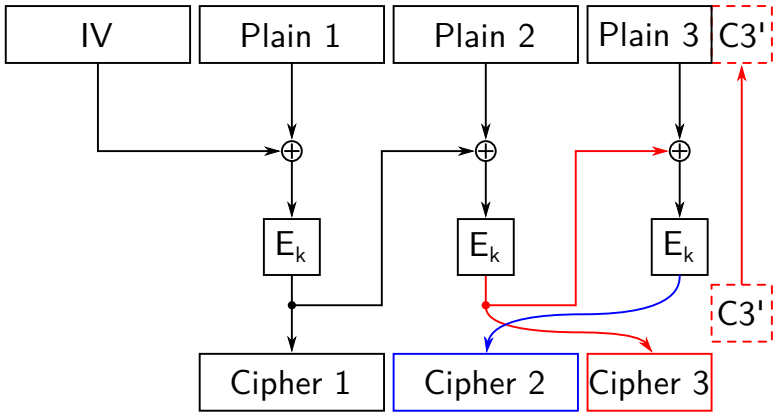


Figure 3.8: Cipher-text stealing prevents cipher-text expansion by rearranging the blast two blocks.

The protocol supports two modes, *authenticated encryption* and *authentication*. In order to provide IND-CPA, a constantly changing initialization vector (IV)

composed of a counter and most of the header of a packet, is used for the encryption mode. The length of the IV is 8 bytes. The authors accept the fact that IVs will repeat, this is the reason why a block cipher and CBC-MAC were chosen. To address that they suggest to change the key after enough messages have been processed.

3.5.3 MiniSec

MiniSec [54] is based upon TinySec and it uses offset codebook (OCB) mode [47, 69, 70] to reduce the number of encryptions passes over the plain-text in order to obtain authentication and confidentiality. OCB works by generating a cipher-text C from the plain-text message M, a key K and a nonce N. The cipher-text is also used together with the plain-text and the key to generate a tag τ (Figure 3.9). The reverse process is used to obtain the plain-text and to recompute the tag in order to validate it. To obtain C, the offset Δ is first calculated by using different parts of N. The plain-text M is then divided into m blocks, according to the block-size of the underlying encryption algorithm. Each block is encrypted and *xor*ed with Δ. For each of the blocks Δ is updated by *xor*ing it with a value initially derived from the zero vector $\mathbf{0}$. The tag is the τ most significant bits of the encryption of the offset and the running *xor* of all the message blocks. Optionally, some associated data AD can be added to the tag. To do so AD is divided into blocks and each block is encrypted in a similar way to M. The resulting value $Auth$ is then *xor*ed with the tag.

MiniSec provides unicast replay protection through a counter shared between the sender and the receiver. The counter is updated each time that a message is received and messages with a counter value lower than the current one are discarded. In order to save energy a technique called *last bits optimization* is used: instead of exchanging the whole counter in every packet, only the last x bits are sent. This allows the receiver to update its counter upon receiving a legitimate message, as long as the number of consecutive dropped packets is less than 2^x.

MiniSec can also provide replay protection in a broadcast scenario. Two meth-

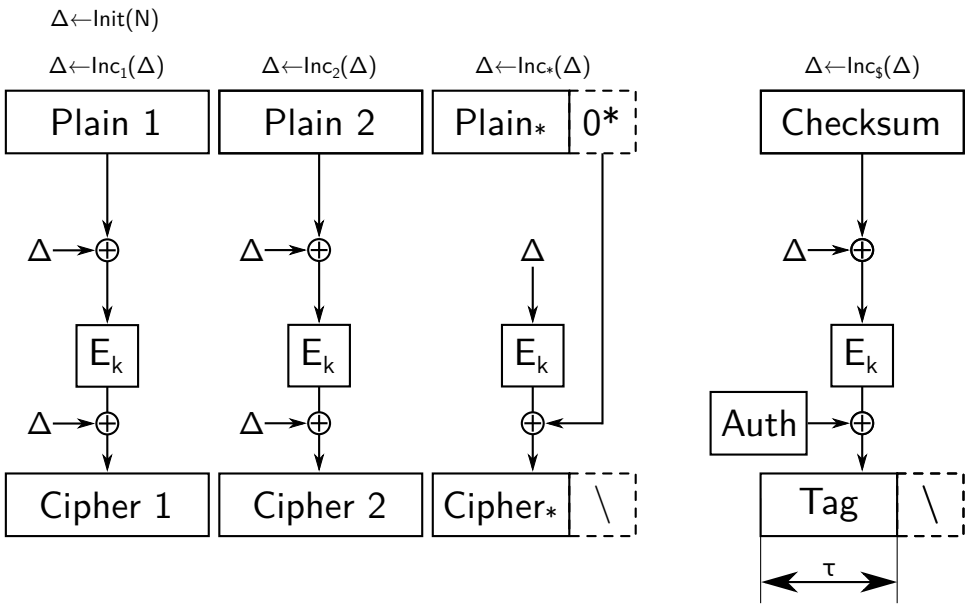


Figure 3.9: Offset Codebook Mode provides both authentication and encryption at the same time. It is also possible to include associated data (Auth) which will be authenticated but not encrypted. [47]

ods are used for this: a sliding-windows approach to defend within a certain vulnerability window, and a Bloom filter approach used to defend against attacks within the window. The former divides time into epochs and make nodes agree upon the current epoch and uses the epoch number as a nonce. To compensate for network latency, two consecutive epochs are considered and decryption is attempted twice. This allows an attacker to replay a message within the same epoch and for a constant portion δ (depending upon network latency and synchronization errors) into the following one. By setting the sliding-window to the length of one epoch plus δ, no replay is possible.

A second approach uses a counter and two Bloom filters, one for each of the last two epochs. The counter is used together with the node *id* and the epoch number as the nonce and reset at the beginning of each epoch. Whenever a packet is received, the receiver checks whether or not it decrypts to meaningful data. If it does the packet must belong to one of the last two epochs. If the packet is valid,

the epoch number is used to check whether the packet is already in the Bloom filter for the corresponding epoch. In case of a hit the packet is considered replayed and discarded, in case of a miss it is considered genuine and it is accepted and added to the filter. Because of the nature of Bloom filters and the fact that they can produce false positive there is the possibility that e genuine packet is deemed replayed and thus rejected.

Finally thanks to the counter used both in unicast and multicast mode, MiniSec can provide weak freshness and semantic security.

3.5.4 SenSec

SenSec [50] is a transparent link layer security scheme based upon TinySec. It provides only one mode with both authentication and encryption. SenSec uses a partially randomized IV that is 8 bytes long and a modified version of Skipjack called Skipjack-X. This version uses a technique similar to DES-X [44] where given an 80-bit key K and two additional 64-bit keys K_1 and K_2, then the encryption of a message M is obtained as $K_2 \oplus \text{Skipjack}(K, K_1 \oplus M)$. The key length for Skipjack-X is increased from 80 to 208 bits. However, it is possible to obtain an attack that reduces the effective length of the key to 111 bits [48].

SenSec also introduces and uses a modified version of CBC called CBC-X (Figure 3.10) which provides both authentication and encryption at the same time. A stealing technique is used in order to avoid data expansion.

Three level of keys are used within the system: global keys, cluster keys and sensor keys. Each key is pre-generated and pre-loaded.

We have discussed about security in both regular and EH WSNs. However, the most common solutions in literature do not provide anything specific to, nor they differentiate between battery and EH powered approaches whatsoever. This is what we are going to focus on in the next chapter.

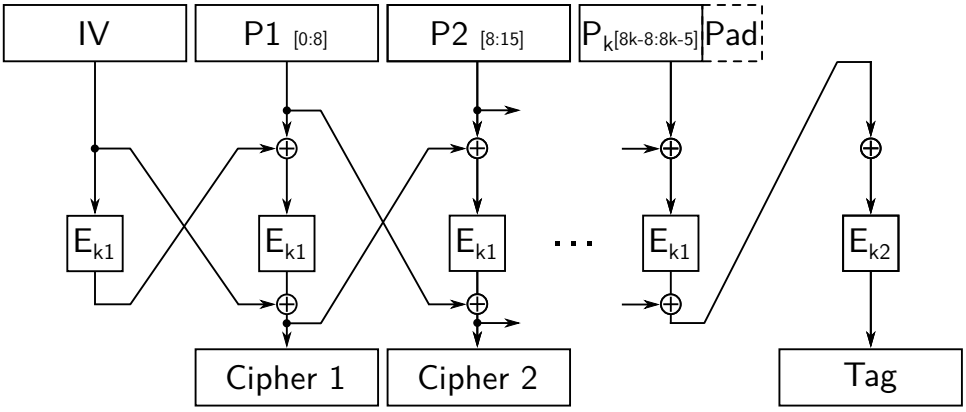


Figure 3.10: Scheme of CBC-X with code stealing [50].

Never trust a computer you can't throw out a window.

Steve Wozniak

4

Designing a Secure MAC Protocol: ODMAC

After having introduced security in WSNs and having extensively discussed about security in the data link layer, we will now turn our attention to a specific MAC protocol, namely the on-demand medium access control (ODMAC).

ODMAC [24] has been developed specifically to accommodate the needs of EH-WSNs. Its design and theoretical evaluation work are not a contribution of this dissertation and belong to the respective authors [24, 25]. In Section 2.2 we introduced EH-WSNs and discussed how they are fundamentally different from regular WSNs. Their theoretical infinite lifespan poses unique challenges. Furthermore, EH-WSNs are also characterized by *spatial inconsistencies*: depending on the particular source of energy being scavenged it is not uncommon to find completely different energy situation between different portion of the network, independent

of their physical correlation. For example imagine a solar-powered EH-WSNs with nodes laying on either sides of a wall. The nodes might be well in range and able to communicate, but depending on the time of day one side of the network could be in the shadows and unable to harvest energy. Depending on the topology and the application being run, this might impact availability or even disconnect the network making this a concern of the MAC protocol.

In general, all the challenges found in EH-WSNs, be they unique to this setting or inherited from regular sensor networks, must be addressed in a different way, accommodating for the requirements and the characteristics of EH devices.

4.1 Protocol Description

The main tool that ODMAC has in order to deal with the aforementioned challenges, is to allow each node to independently choose its own DC and adjust it according to different parameters like the harvesting rate of a node or the requirements of the application. Thanks to that, a trade-off between power and performance can be obtained. For example in case of scarce energy the DC can be decreased to give time to the node to harvest more and gather enough to survive a communication. On the other hand, when energy is abundant the DC can be increased and as a result also the performance of the network will increase: more packets exchanged will translate to increased throughput and decreased delay.

To achieve this, the protocol relies on the RI paradigm, which not only has proven to be more energy efficient than its counterpart SI [51], but is also a good match for DC adaptation. A node running ODMAC has two different DCs one for exchanging messages and one for sensing purposes. These two DCs are respectively called the *beaconing* and the *sensing* duty-cycle.

As the name of the paradigm suggests, the message transmission starts from the receiver. Whenever a beaconing period elapses and a node r ready to receive data enters the active state, it will perform a CCA to determine whether or not there is an ongoing transmission already happening. If the channel is available, r will transmit a beacon b manifesting its intentions to receive a packet and it will then

start to listen to the channel for incoming packets for a fixed amount of time.

Similarly when a sender *s* enters the active state because a sensing event has occurred, it will start listening to the channel for an *appropriate* beacon. This is a beacon that satisfies specific predefined conditions such as moving the packet closer to its destination. Should such a beacon be received *s* will immediately transmit its packet and go into sleep mode. In the upcoming wake-ups, *s* waits for a new beacon from *r* which will work as an ACK for the packet sent previously.

Whenever data is received by a node, if the node itself is not the final recipient (as it will happen most of the times in a multi-hop network), a forwarding procedure will begin. This is identical to the sensing and transmission operation described above, with the exception that instead of being generated locally, data is obtained from another node.

A more "implementation oriented" summary of ODMAC can be seen in Figure 4.1 which describes the protocol as a finite state machine.

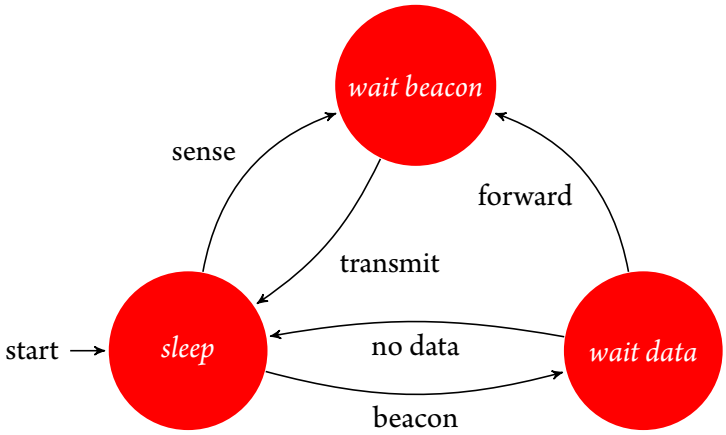


Figure 4.1: ODMAC described as a finite state machine.

Being ODMAC specifically designed for EH-WSNs, the guiding principle is almost always *sustainability*, more specifically a node adapts is DCs to remain operational with the current energy situation, aiming to achieve what we introduced before as ENO state. The way this is obtained is through a feedback loop where a node is able to monitor its current energy level and initiate a communication when

the value is high enough.

4.1.1 Opportunistic Forwarding

One of the features of ODMAC is *opportunistic forwarding*. The purpose of this is twofold, it tries to relieve the hub nodes from some of the high traffic that they would normally experience, and it helps to increase the overall performance of each link.

Assuming a network with fixed topology, the result of running a networking protocol will be a (locally distributed) list of routes which tell each node to which neighbor to send a packet to, based on its final destination. If we consider the peculiar traffic pattern of a WSN, we already know that most of the packets move from the nodes towards the sink. This means that, except for node failures, the routes will be static and the minority of nodes that have been highly rated by the routing protocol will experience most of the traffic and therefore they will consume more energy on average. This has a negative impact on the network, especially considering that within ODMAC when a node consumes more energy it will decrease its DC and slow down the beacon transmission rate.

In order to address this problem, opportunistic forwarding allows each node to transmit to a set of receivers that allow the packet to move closer to its final destination, even if the route used is sub-optimal. The way a receiver is selected is on a *first come first served* basis, the first beacon received that originates from a node in the set, is answered to by the sender.

From the receiver standpoint the traffic will be distributed more evenly across all the possible receivers and high-energy nodes will be automatically selected thanks to their increased beaconing rate. The sender will also obtain benefits in the form of a reduced *idle listening* time. In order to receive a beacon the radio of the sender must be turned on in listening mode. The longer the radio is used, the more energy is consumed. By answering to the first appropriate beacon, a sender will decrease this period of time, thus saving energy.

Furthermore, by opportunistically selecting beacons, the average amount of time

that a packet spends waiting to be relayed to the next hop is also decreased. This means that the overall delay of the network is decreased, a factor that mitigates the use of what could be sub-optimal paths.

4.1.2 Altruistic Backoff

Another feature of ODMACs is altruistic backoff (AB), a *collision avoidance* technique. Collisions are physiological in MAC protocols and ODMAC is not different. Whenever two nodes that want to transmit to the same receiver are awake at the same time and obtain a beacon from said receiver, a collision is highly likely to happen. Both nodes will try to use the beacon and occupy the channel at the same time. The most commonly used technique to deal with collision is random backoff (RB), where each node chooses a random value before transmitting and "backs off" for that amount of time before performing the actual transmission, which then takes place only if the channel is available after the delay.

In a RI protocol in general, and within ODMAC in particular, RB is costly because it prevents the collisions at the latest possible moment, when everything but the actual transmission already happened, including waiting for and receiving a beacon. The idea behind AB is for a node to give way to other nodes waiting for the same beacon. This is done through an altruistic backoff request (ABR) packet. When transmitted by a sender s, the goal of this packet is to inform other potential senders about what kind of receivers s is waiting for. If any other node is waiting for beacons from the same receiver, they will backoff leaving the channel to s.

The procedure is repeated by each node when it wakes up so that the last node issuing an ABR is the one obtaining the channel.

Contrarily to RB, the ABR transmission is done as soon as a sender wakes up and before a beacon is sent and received. As soon as a node receives an ABR for a common receiver it goes immediately back into power saving mode, allowing for additional energy saving.

4.1.3 Layer-based Anycast Routing

One more feature that can be found in ODMAC is layer-based anycast routing (LAR). Strictly speaking this is a routing extension rather than a link layer functionality, and it shows how ODMAC is supporting a cross-layer approach.

LAR has two phases, a set-up phase and a steady phase. During set-up each node is assigned a *layer* value, which is the smallest number of hops between itself and the sink node. The value of each layer is calculated through a distributed breadth first search (BFS) rooted at the sink, where each node sets as its own layer the value received by neighboring nodes, incremented by one, provided that it is less than the current calculated value. Each node will then transmit its layer for the information to propagate.

During the steady phase, each node considers appropriate the beacons with a layer value less than its own (also the value equal to the layer of the node are accepted if intra-layer routing is allowed) and uses them to relay messages. If after a predefined period of time a node does not receive any appropriate beacon, the layer number is reset and re-initialized according to the value of the beacons received. A similar approach is used by nodes deployed at a later time to join the network when is already operational or to reorganize it in the case of failures. This functionality copes very well with opportunistic forwarding.

4.2 Security of ODMAC

So far we have been discussing about the importance of the data link layer and the MAC protocols, even described some security suites for them and introduced a new protocol specific for EH-WSN, namely ODMAC. The natural question that arises now is: *"How secure is ODMAC?"*. This is what we will address now.

Given the current state of the protocol and according to what has been introduced so far, the answer is: *"Not very"*, let us discuss the potential weaknesses of ODMAC.

Based on the definition of security that we gave in Chapter 3, the minimum

properties that we would like to have are: *confidentiality, integrity* and *availability*. So far ODMAC does not provide any of this. Messages are sent in the clear and there is no control over their authenticity, allowing an attacker to eavesdrop the communication, intercept messages and forge new ones. This would make pretty much all the attacks presented in Section 3.2 possible.

To address this, we developed a security suite inspired by TinySec [43]. The security suite has four modes of operation: *no security, authentication, encryption* and *authentication+encryption* which can be chosen on a per-message basis, allowing for full customization from the user. When both authentication and encryption are chosen, they are composed using the secure *encrypt-than-MAC* paradigm [46].

The scheme supports encryption algorithms with 64-bit blocks and 80-bit or 128-bit keys. In our proof of concept implementation we have used Skipjack [63] which requires 20 B of RAM and around 6.5 KiB of ROM. Despite its age, Skipjack continues to prove secure for an 80-bit key algorithm [45], the most successful attack so far is an impossible differential attack on 31 of the 32 rounds yielding a result marginally faster than exhaustive search [6]. Considering that the national institute for standards and technology (NIST) has proposed to phase out the use of 80-bit keys by 2015, it is a good idea to turn the attention to algorithms supporting 128 bits keys. Depending on the application and the implementation, different ciphers can be used [48]. Piccolo [75] and TWINE [78] are good candidates for software implementations respectively requiring 91 B and 23 B of RAM and 2.5 KiB and 2.2 KiB of ROM. For hardware implementation the best candidate is PRESENT [8] which uses 1886 gate equivalent (GE) and has also been included in the standard for lightweight cryptographic methods by the international organization for standardization (ISO) [30].

Encryption is carried out using CBC mode with cipher-text stealing to avoid last block message expansion, while authentication is done with CBC-MAC. This allows to use one encryption algorithm to perform both operations. It is important to highlight that a key for the system in *authentication+encryption* mode is actually a pair of keys, one for each operation. Authentication codes and encryption are checked and re-computed at each hop. This has the advantage of intercepting ma-

liciously or fortuitously malformed packets as early as possible, avoiding to waste energy to route them to their final destination only to discard them there.

A possible extension to the scheme is to use an authenticated encryption (AE) mode. The advantage of this technique is that it is possible to obtain both authentication and encryption at the same time, without having to run the algorithm twice. The highest performance algorithms are OCB (used in MiniSec [54]) and Galois counter mode (GCM), with the former having the better performance [47]. Unfortunately OCB is patented and could not be used freely until recently (9^{th} January 2013), when a free license has been issued for open-source non-commercial application. The algorithm is still not free for commercial applications. The other mode, GCM, is notoriously cumbersome to implement correctly. A comparison between different implementations in TinySec can be found in [41] where CBC and GCM are analyzed in conjunction with both the advanced encryption standard (AES) and Skipjack. The results show that GCM in combination with AES obtains a 12% increase in energy consumption, a 28% increase in RAM usage and a 35% decrease in throughput compared to the original implementation of TinySec. While constantly outperforming CBC in combination with AES, it is still a considerable decrease in performance, justifiable only if the application requires both authentication and encryption without any differentiation.

Thanks to the addition of our scheme, ODMAC achieves much better security properties while still maintaining a low-resource profile and remaining suitable for EH-WSNs.

One final component included in ODMAC is the receiver authentication protocol (RAP), a protocol specifically designed to prevent a new attack that we call *beacon replay attack*. We will introduce and discuss this in Chapter 5.

The trouble with quotes on the Internet is that you never know if they are genuine.

Abraham Lincoln

5

Beacon Replay Attack

In Section 3.2, we have introduced several well-known attacks, mostly for regular WSNs. However, we have reiterated many times how EH-WSNs follow different rules and require different solutions. With respect to the link layer, we believe, these solutions are strongly connected to RI protocols and that ODMAC is a good candidate to serve as a customizable platform that can be tweaked according to the application requirements. In our working with these protocols, we have discovered that ODMAC and RI protocols in general are susceptible to a powerful attack, the *beacon replay attack*.

5.1 Classic Replay Attack and Protection

As presented before, the replay attack [15] is a well-known attack where a previously sent piece of information is recorded and re-transmitted at a later time,

unmodified. In WSNs this constitutes a significant threat since replay attacks are very commonly used as an essential building block for more complex and effective attacks such as sinkhole and blackhole attacks [42].

For example a more involved attack that can be mounted by replaying messages is path denial of service (PDoS) [14]. This is an attack where a whole path from one sensor node to the BS is filled with bogus packets. Given that a WSN is typically structured as a tree rooted at the BS, not only the node at one end of the attacked path can not use the communication medium, but also all the nodes *along* the path are prevented from forwarding their own messages. Furthermore, depending on the specific application that is being run on top of the network, replayed data messages could pose different kind of threats according to their specific meaning.

Other previous works have addressed and mitigated replay attacks. The most common solution is to make each packet unique by means of adding either a counter or a timestamp. Timestamps are usually harder to implement because they require an agreement between the sender and the receiver which, in turns, translates to a global agreement for forwarded packets. This is an expensive property to achieve, and depending on the protocol might even be discouraged. For example the goal of ODMAC is to allow each node to independently regulate its own DC. An alternative is represented by monotonically increasing counters that are generally fed to a cryptographic message authentication code (CMAC), making sure that each message is unique.

5.1.1 Beacon Replay Attack in the Receiver-Initiated Paradigm

The beacon replay attack can be mounted by capturing and replaying *beacon* packets. This might sound like a pedantic redefinition, but as we will show it constitutes a different kind of attack in and of itself. We know that beacons manifest the availability of a particular node to receive a message. Among other optional control values, they contain the identity of their creator which is the main piece of information needed to determine whether or not a specific beacon can be used by a

potential sender, and who to send the packet to. By replaying beacons containing good identities (typically from a routing point of view), it is possible to obtain several harmful effects and mount other attacks.

First of all, an attacker could flood the channel with these frames, pretending to be an extremely active receiver and trying to accumulate as many data packets as possible. By definition this is a sinkhole attack. After the acquisition, packets can be completely dropped thus performing a blackhole attack. A subtler possibility is to implement a selective forwarding attack, where packets are not dropped indiscriminately, but rather according to their source. This yields a harder to detect and yet still very effective attack.

By replaying beacons containing different identities to the same sender nodes, it is possible to have a sybil attack. This could lead to routing paths to become invalid, or even nodes that are physically not within range one another, to be led to believe so; turning this into a rudimentary one-man wormhole attack.

One last meta-attack, specific to duty-cycling wireless networks, is what we call the *sleepwalker* attack. The idea behind it is that if the legitimate owner of a beacon receives a message with its own identity, it can easily detect that there is an ongoing attack and warn the rest of the network. However, all the previous attacks can be deployed by a malicious node that is within range of the attacked node simply by exploiting the notion of DC. Beacons can be collected from a node and replayed in the same neighborhood when the original sender is asleep. In this way a malicious node can effectively masquerade itself as another node.

An additional problem with the beacon replay attack is that the techniques that we briefly introduced before and that are normally used to counter regular replay attacks do not apply in this scenario. One of the advantages of an RI approach is the fact that no synchronization is needed for the protocol to operate. Timestamps, in order to be meaningful, require some form of clock synchronization among the nodes. This usually comes for free within protocols that use synchronized DCs, but is a costly feature to obtain in RI protocols.

The other common alternative is the use of counters and session numbers. The latter are random non-reusable numbers that uniquely identify a particular mes-

sage, or in this case a beacon. In order to check if a received beacon is fresh or replayed, a table of all the previously used session numbers should be kept. Given the highly constrained resources of a sensor node, and the fact that there should be such a table for each one of the neighboring nodes, this solution is inapplicable. One way of simplifying this mechanism is to replace the random number with a monotonically increasing counter. This eliminates the need of having to store a whole table, only the latest value is needed. Upon receiving a message the new counter value can be compared against the last received one and if newer (i.e., the received value of the counter is bigger than the previous one) it will be accepted and discarded otherwise. The reason why this mechanism does not work with an RI protocol is the following. Beacons are sent with a periodic cadence, which is typically randomized in order to minimize collisions. If we also consider all the neighboring nodes, from the point of view of a specific node, the arrival time of a beacon is virtually uniformly distributed. This means that there is no way for a sleeping node to know how many beacons were sent between the current and the previous active period, allowing the attacker to replay beacons that were not received by sleeping nodes.

Moreover, a downside of both timestamps and counters, is that some extra information has to be sent with every beacon, even the ones that will never be received, because all the other nodes are asleep. This constitutes a costly overhead.

Lastly, despite the fact that CMACs can be used to authenticate beacons, they cannot prevent a replay attack. All that can be guaranteed upon receiving a beacon whose authentication tag correctly matches, is that the at some moment in time that beacon was genuine, created by a legitimate node and intended for another legitimate node. However, it is not possible to establish whether or not the beacon that has just been received is actually *that* beacon.

For all these reasons, we have designed RAP, a novel authentication scheme specifically designed to detect and prevent the beacon replay attack in receiver-initiated MAC protocols.

5.2 Receiver Authentication Protocol (RAP)

RAP is a challenge-response authentication protocol that aims to authenticate receivers, i.e., the beacon transmitter, in an RI data transmission, securing the communication in general. RAP is compatible with and can be used on top of every MAC protocol that follows the RI paradigm, essentially securing the whole class of protocols from beacon replay attacks. Moreover, RAP can and should be used together with security suites that provide other security features such as data integrity and confidentiality.

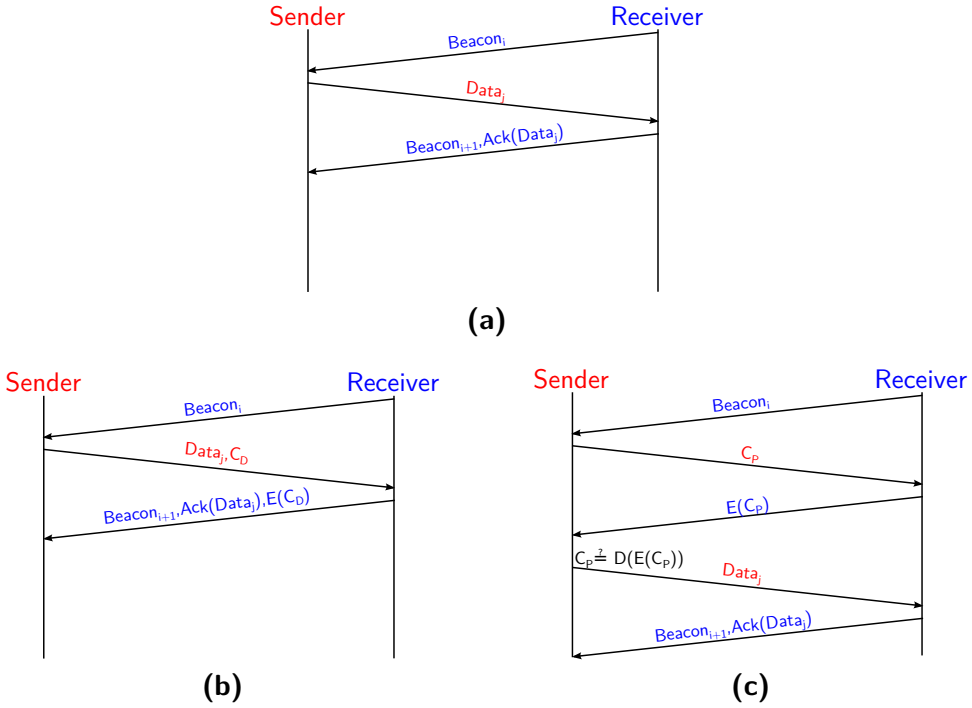


Figure 5.1: A typical receiver-initiated protocol (a), RAP-D (b), RAP-P (c).

RAP has two modes of operation as shown in Figure 5.1, namely *detection* and *prevention* mode. In a nutshell, the receiver authentication protocol detection mode (RAP-D) is a low overhead scheme that aims at detecting an intruder that replays beacons without stopping it from doing so. The receiver authentication protocol

prevention mode (RAP-P), on the other hand, is a more costly scheme that stops the attack altogether. As described in the following sections, the key difference between the two modes is the timing of the challenge-response message exchange. In RAP-P, the challenge phase takes place *before* the data transmission. Thus, the sender transmits the data packet only if the receiver is authenticated. The low overhead nature of RAP-D, on the other hand, is maintained by piggybacking the challenge and its response on top of the frames normally exchanged in the MAC protocol. In other words, the authentication of the receiver takes place *after* the data transmission (thus, the attack is not immediately prevented). Having energy efficiency as a primary system priority, the idea is that a node normally operates at the low overhead detection mode and switches to the expensive prevention mode only if necessary.

5.2.1 Detection Mode (RAP-D)

RAP-D is aiming at detecting beacon replay attacks with low communication overhead. The protocol works as shown in Figure 5.1b. Consider that a sender node S wants to transmit some data to a receiver node R. After R broadcasts a beacon, S answers back with a data packet and a challenge value C_D. On its following beacon, R acknowledges the reception of the data packet, and attaches the encrypted version of the challenge $E_{k_{RAP}}(C_D)$ using the protocol specific, shared key k_{RAP}. At this point R can validate the response to the challenge by decrypting and checking it against its original value. Should these two values not match, then R can conclude that the initial beacon was not genuine.

RAP-D adds a minimal overhead in the whole communication scheme, as the challenge and the response are piggybacked on top of a regular message exchange. Furthermore, if the challenge C_D is transmitted as part of the payload and encrypted with it, its size can be relatively small without risking of increasing the chances of success of a space exhaustion attack (see Section 5.3.2).

5.2.2 Prevention Mode (RAP-P)

RAP-P is aiming to prevent the beacon replay attack at the cost of an increased overhead. In particular, the challenge-response messages are exchanged before the data transmission, in order to distinguish the legitimate from the replayed beacons. The protocol works as shown in Figure 5.1c. Instead of sending the data right after a beacon, S sends out a longer challenge C_P, and awaits for its encrypted version $E_{k_{RAP}}(C_P)$ from R. Only if the received value decrypts correctly (i.e., matches against C_P), then the data are sent. This scheme is more expensive because it requires two additional messages to be exchanged. Additionally, the size of the challenge needs to be significantly larger than the detection mode to prevent space exhaustion attacks.

5.2.3 Transition Policies

Depending on the security goal of an application, RAP can be configured to switch between the two modes, using several policies. If the application cannot tolerate a few beacons getting replayed, the protocol should always operate in prevention mode for maximum security. In the opposite case, the detection mode should be the default to promote energy efficiency. Here, the transition from RAP-D to RAP-P should be done after a defined number of challenge mismatches. This number should be configured accordingly to account for channel errors. Furthermore, the intruder detection may trigger an alarm that can be piggybacked onto data packets and beacons in order to warn the neighboring nodes and the sink of an ongoing attack. The transition back to detection mode can be done either automatically or manually depending on the desired level of security. In cases of high security requirements, it may be desired that RAP-D is re-activated manually by the system administrator only after an investigation. An automatic transition to RAP-D, can be done after a predetermined number of successful challenge matches. To avoid the exploitation of this transition policy, the threshold value can be exponentially increased each time a new replay attack is detected.

5.3 Verification and Analysis

5.3.1 Verification with OFMC and ProVerif

In order to formally verify RAP, we modeled it using the Alice and Bob (AnB) language. AnB [59] is a specification language based on the popular Alice-and-Bob notation for security protocols. Besides giving us a way to describe the protocols of interest in a succinct way, AnB is also a formal language with an unambiguous semantics of the honest agents, the intruder, and the goals of the protocol. The semantics of AnB is defined by translation to infinite-state transition systems and its attack states, described in the AVISPA intermediate format (IF) [2]. The IF can be directly read by several tools, such as on-the-fly model checker (OFMC) [4]. We also manually translate AnB specification to the abstraction-based tool ProVerif [7]. The main idea for using two tools lies in their complementary strengths. OFMC is effective in finding attacks, but can verify a protocol only for a bounded number sessions; on the other hand ProVerif abstracts from the concrete search space, sometimes producing false attacks (especially for replay-protection goals), requiring adaptations of the specification. Therefore, verifying the protocols with different approaches gives a higher confidence.

The core of the AnB specification is the definition of the behavior of each role of the protocol when it is played by an honest agent, namely how this agent decomposes the messages it receives (and what parts of a received message it can actually check), and how the agent composes outgoing messages based on its initial Knowledge and the previously received messages. Here, all variables that do not appear in the knowledge section of the AnB specification are values that are *freshly* created by the agent who first uses them. For instance in the detection protocol RAP-D, S freshly creates the challenge C and the data $Data$. For the full details of the AnB semantics we refer to the original paper [59].

The standard intruder model of AnB is the common Dolev-Yao intruder [21] who controls the entire communication medium, it can arbitrarily overhear, send and even intercept messages. This is clearly inspired by communication in wired

networks. We have argued before that for WSNs this model is fitting for some aspects, but unrealistically strong for others: an intruder may not control all locations spanned by the WSN and also it may not be able to hear a message when it is blocking it (e.g., by jamming). However, verifying the protocol under such a strong intruder gives higher confidence in this particular scenario.

Moreover, unless explicitly excluded in the specification, the intruder can also play as a legal participant of the protocol. In the case of WSNs, this amounts to modeling compromised or intruder-controlled nodes. These dishonest nodes do not need to comply with the protocol, but can send whatever messages the intruder can compose from its knowledge. The initial intruder knowledge is determined also by the knowledge section of the AnB specification: for each instance of a role that the intruder is playing, he gets the associated initial knowledge. For example, consider in the RAP-D protocol a session where S is played by honest agent a and R is played by the intruder i. Then the intruder gets the knowledge of R under this instantiation, i.e., $a, i, mac, sk(a, i)$, and thus he has the shared key needed for communicating with a.

Furthermore, we use authentication goals which correspond to injective agreement ad introduced by Lowe [52]. For the concrete example of the goal *S authenticates R on R, C* used in raps, as soon as R learns the fresh challenge C, it produces (in our model) an auxiliary event $witness(R, S, C)$ formalizing the intention to run the protocol with S and to use C. When S successfully finishes her run of the protocol, she produces also an event $request(S, R, C)$ to formalize that she finished the protocol, apparently with R and using challenge C. It counts as an attack if a trace contains more request events than corresponding witness events, i.e., when S either believes in receiving something from R that R actually has never sent, or if S is tricked into accepting something more times than R actually sent.

Finally, we use the channel notation introduced by Maurer [57], which is supported by the AnB language (for the formal definitions in AnB see [61]). Informally $A \bullet\!\!\rightarrow B$ means that A sends a message *authentically* to B (so B can be sure it really comes from A and was meant for B), $A \rightarrow\!\!\bullet B$ means that the message is sent confidentially (so A can be sure only B can receive it), and $A \bullet\!\!\rightarrow\!\!\bullet B$ means both

authentic and confidential transmission. We use this notation to abstract from how the transmission of the actual data is organized, i.e., how authentication and confidentiality is achieved if they are desired. In fact, this problem is orthogonal to the replay-protection for the beacon that we study here, and the channel notation allows us to abstract from that. We note however that the actual realization of such channels (e.g., by CMAC and/or encryption) needs to compose with our replay-protection, as explained in [61]. In short, if both our replay protection and the secure channel implementation use symmetric encryption with the same shared key, this can lead to misunderstandings in the WSN that may be exploitable. If they use however different keys (possibly derived from the same root key) this is prevented and the composition is sound.

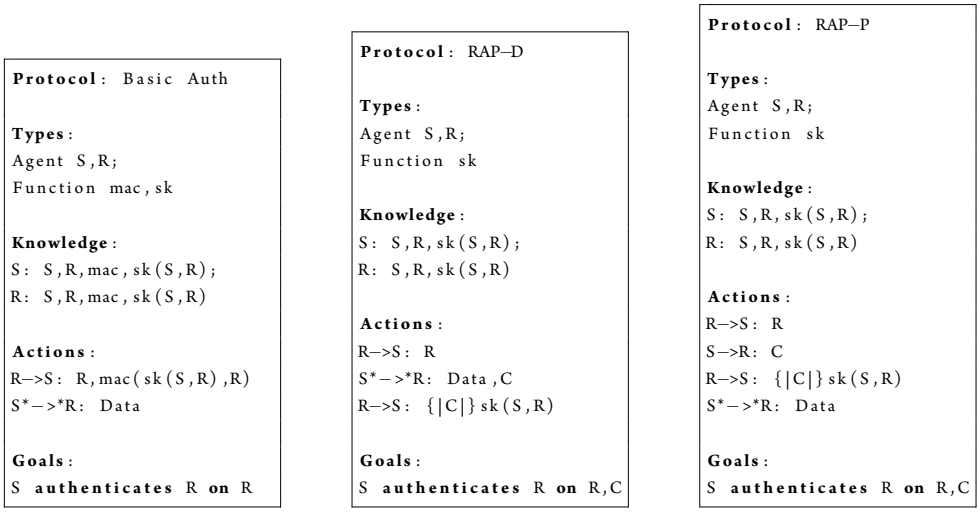


Figure 5.2: The protocols used with OFMC described in AnB notation. A basic authentication model (a) is only enough to prevent beacon forgery. RAP-D (b) and RAP-P (c) are not affected by beacon replay attacks.

In Figure 5.2 it is possible to see how we modeled RAP using the AnB notation [59]. It should be noted that we decided to strip down the protocols in order to focus the attention on the beacon replay attack, hence we kept only the messages relevant in this sense. Furthermore, we also decided not to include the basic

version of the paradigm which does not include any form of authentication. This protocol is essentially modeled like the basic version (Figure 5.2a) but without a CMAC for the beacon. This yields the trivial attack of beacon forgery due to the complete lack of authentication.

In the case of basic authentication (Figure 5.2a), OFMC can detect the beacon replay attack, shown in Figure 5.3, within a few seconds. For the intruder i it is simply enough to store a previously received beacon and replay it to a victim node in order to receive the data. Another interesting fact is that by adding the *weakly* clause to the authentication goal, hence turning it into Lowe non-injective agreement [52], no attack is found. This helps to build confidence in the model and its correctness.

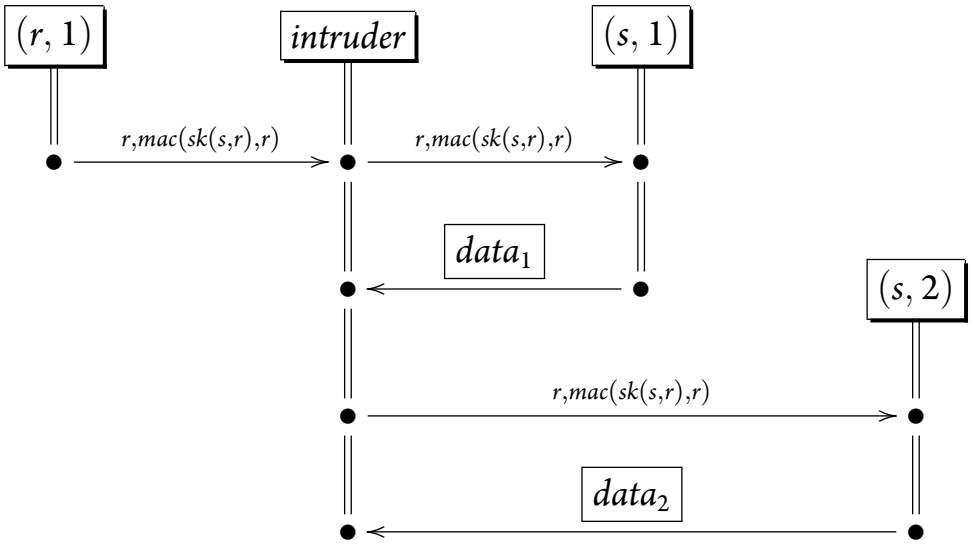


Figure 5.3: Trace of the beacon replay attack found by OFMC in the basic version of a receiver-initiated protocol. The intruder intercepts the beacon and forwards it to the sender (s) completing a session. subsequently, in a new session, the same beacon and its associated CMAC are replayed, obtaining new data.

When running OFMC on RAP-D and RAP-P, we can verify them for three sessions , without any attack. Note that in each session, OFMC considers all possible instantiations of the roles with concrete agents, both honest and the intruder.

Thus, whenever a protocol is verified for a given number of sessions, then there is no instantiation of the roles for these parallel sessions that can lead to an attack. As a rule of thumb, attacks are usually detected within two sessions.

ProVerif computes on first-order Horn clauses [35] that represent an over-approximation of the reachable events and messages the intruder can ever learn. There is therefore no notion of time-line, posing some difficulties for the analysis of replay, even though ProVerif offers the notion of *injective* events for this purpose. In order to experiment with different settings, we used the AIF framework [60] built on top of ProVerif, allowing to specify a state-transition system with a number of sets of data. In this particular case we can define for each agent the set of challenges that are sent out and have not been responded to, as well as those that have been responded to (and are therefore *used*). The AIF framework also allows for producing the Horn clauses for a different tool (on which ProVerif was originally based): the automatic first-order theorem prover SPASS [85]. It is therefore without extra cost to check the verification also with SPASS. Both tools successfully verify the protocols.

5.3.2 Space Exhaustion Analysis

In this section we conduct a space exhaustion analysis on RAP. Specifically, an attacker can passively monitor the communication of legitimate nodes and collect pairs of challenge and response messages. In this way, the attacker can gradually build a dictionary that can be used to bypass RAP. The size of such a dictionary is a direct indication of the resilience of the protocol against space exhaustion.

When RAP is in prevention mode, an attacker can trivially map the challenge to the respective response, as they are both distinct messages. Thus, the size of each word $D_{\text{RAP-P}}$ in the dictionary is equal to the size C_P of the challenge in bits, translating to $2^{D_{\text{RAP-P}}}$ words.

$$D_{\text{RAP-P}} = C_P \tag{5.1}$$

When RAP is in detection mode, we aim at a small challenge to keep the overhead

low. However, the dictionary size can be significantly increased by encrypting the challenge together with the data, using CBC mode. Essentially, CBC hides the challenge within the data, preventing the attacker from mapping it to the response. As a result, a dictionary can only be built by mapping the whole message (that contains both the data and the challenge) to the respective response. Therefore, the size of each word $D_{\text{RAP-D}}$ in the dictionary, which translates to a dictionary size of $2^{D_{\text{RAP-D}}}$ words, is equal to the aggregate size L_D of the data and C_D of the challenge.

$$D_{\text{RAP-D}} = C_D + L_D \tag{5.2}$$

As an attacker can force the system to change the mode of operation, we note that the overall resilience of RAP to space exhaustion is equal to the smallest of the two dictionaries, $D_{\text{RAP-D}}$ and $D_{\text{RAP-P}}$. Furthermore, the sizes of the two challenges, C_D and C_P, which constitute configurable protocol parameters, define the level of security in the same manner the size of a key defines the level of security of an encryption algorithm.

5.3.3 Energy Consumption Analysis

In this section, we attempt to model the energy overhead of RAP and highlight the trade off between security and energy constraints.

Let L_D be the size of a data packet in bits, L_B be the size of a beacon in bits and λ the transmission rate of the radio in bits per second. Additionally, let P_{tx} and P_{rx} be power consumption for transmitting and receiving / listening respectively. First, we estimate the energy consumption for a single packet transmission in the case of not using RAP. For the receiver, R, the energy consumption is estimated by (5.3), where t_G is a time guard during which the radio is turned on while waiting for a answer right after a transmission. The purpose of such a guard is to account for the propagation and the processing delay.

$$E_R^{\text{Default}} = \frac{L_B}{\lambda} P_{\text{tx}} + t_G P_{\text{rx}} + \frac{L_D}{\lambda} P_{\text{rx}} + \frac{L_B}{\lambda} P_{\text{tx}} \tag{5.3}$$

For the sender, S, the energy consumption is estimated similarly.

$$E_S^{\text{Default}} = \frac{L_B}{\lambda} P_{\text{rx}} + \frac{L_D}{\lambda} P_{\text{tx}} + t_G P_{\text{rx}} + \frac{L_B}{\lambda} P_{\text{rx}} \tag{5.4}$$

Note that this energy model disregards the energy consumed while the sender awaits for the beacon, as this source of energy consumption is independent of the security protocol.

In the case of RAP-D, the energy consumption for a single packet transmission, for the receiver (R) and the sender (S), is given by the following formulae.

$$E_R^{\text{RAP-D}} = \frac{L_B}{\lambda} P_{\text{tx}} + t_G P_{\text{rx}} + \frac{L_D + C_D}{\lambda} P_{\text{rx}} + \frac{L_B + C_D}{\lambda} P_{\text{tx}} \tag{5.5}$$

$$E_S^{\text{RAP-D}} = \frac{L_B}{\lambda} P_{\text{rx}} + \frac{L_D + C_D}{\lambda} P_{\text{tx}} + t_G P_{\text{rx}} + \frac{L_B + C_D}{\lambda} P_{\text{rx}} \tag{5.6}$$

In the case of RAP-P, the energy consumption for a single packet transmission, for the receiver (R) and the sender (S), is estimated similarly.

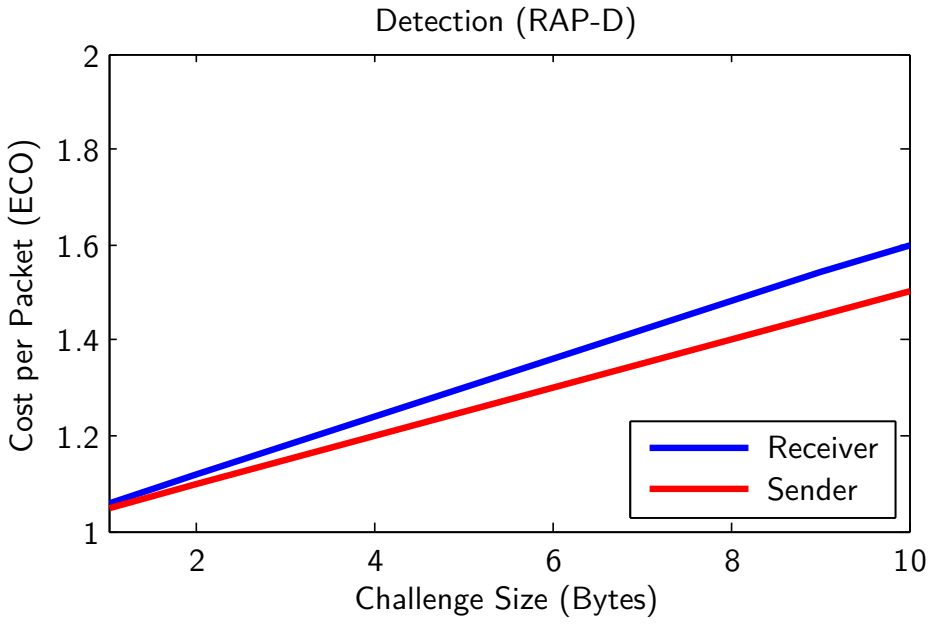
$$E_R^{\text{RAP-P}} = \frac{L_B}{\lambda} P_{\text{tx}} + t_G P_{\text{rx}} + \frac{C_D}{\lambda} P_{\text{rx}} + \frac{C_D}{\lambda} P_{\text{tx}} + t_G P_{\text{rx}} + \frac{L_D}{\lambda} P_{\text{rx}} + \frac{L_B}{\lambda} P_{\text{tx}} \tag{5.7}$$

$$E_S^{\text{RAP-P}} = \frac{L_B}{\lambda} P_{\text{rx}} + \frac{C_D}{\lambda} P_{\text{tx}} + t_G P_{\text{rx}} + \frac{C_D}{\lambda} P_{\text{rx}} + \frac{L_D}{\lambda} P_{\text{tx}} + t_G P_{\text{rx}} + \frac{L_B}{\lambda} P_{\text{rx}} \tag{5.8}$$

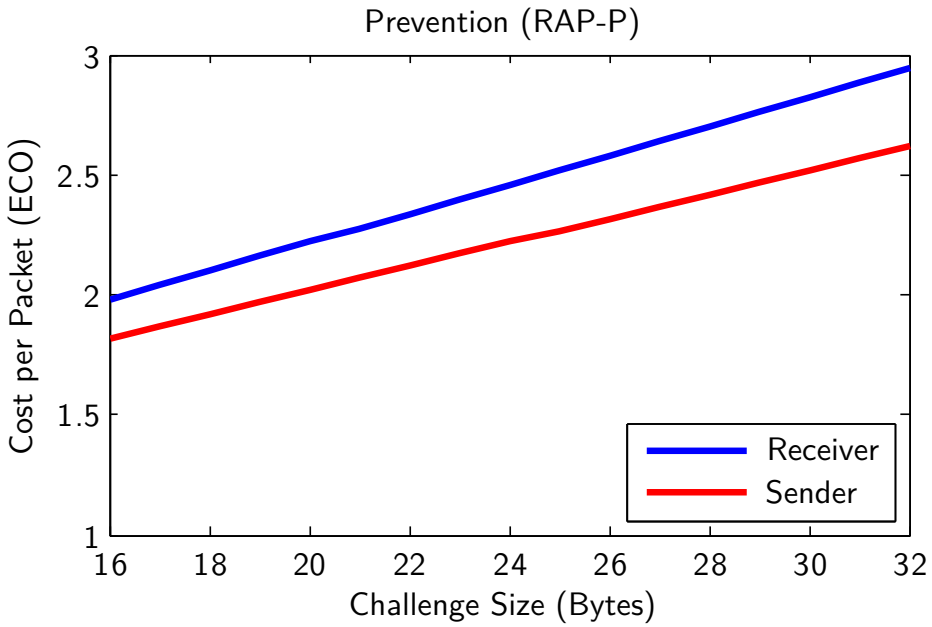
We define the energy consumption overhead (ECO) of a protocol as the ratio of the energy consumption for a single packet transmission (while using the respective protocol) over the case of a plain communication (without using it). The subscript j is equivalent to R for the receiver and S for the sender.

$$\text{ECO}_j^{\text{RAP-D}} = \frac{E_j^{\text{RAP-D}}}{E_j^{\text{Default}}}, \qquad \text{ECO}_j^{\text{RAP-P}} = \frac{E_j^{\text{RAP-P}}}{E_j^{\text{Default}}} \tag{5.9}$$

For the following numerical results, we assume using the *CC2500* radio [39]



(a)



(b)

Figure 5.4: Energy consumption overhead for a single packet transmission for RAP-D (a) and RAP-P (b).

which has the following characteristics: $\lambda = 500$ kbps, $P_{\text{tx}} = 53.8$ mW, $P_{\text{rx}} = 42.5$ mW. Additionally, we consider the following values for the protocol parameters: $L_B = 2$ B, $L_D = 32$ B and $t_G = 10$ µs.

Figure 5.4 shows the cost for a single packet transmission of the two protocols, as defined in Equation (5.9). Notice that the cost of the sender and the receiver increases linearly with the challenge size while the cost for the latter is relatively higher. The difference between them also increases as the challenge size increases.

In Figure 5.5, we compare the cost of RAP-D and RAP-P, showing the low-overhead nature of the former. Particularly, we compare the cost overhead ECO_R for the receiver of the two protocols keeping the same dictionary word size D, as defined in Equations (5.1) and (5.2). Note that the dictionary word size indicates the resilience of each protocol to space exhaustion. In the case of RAP-D, we make sure the value of the challenge is at least 1 B by setting it to $C_D = \max(D_{\text{RAP-D}} - L_D, 1)$. As shown in the figure, the cost of using RAP-P is significantly higher than the cost of using RAP-D for the same level of security.

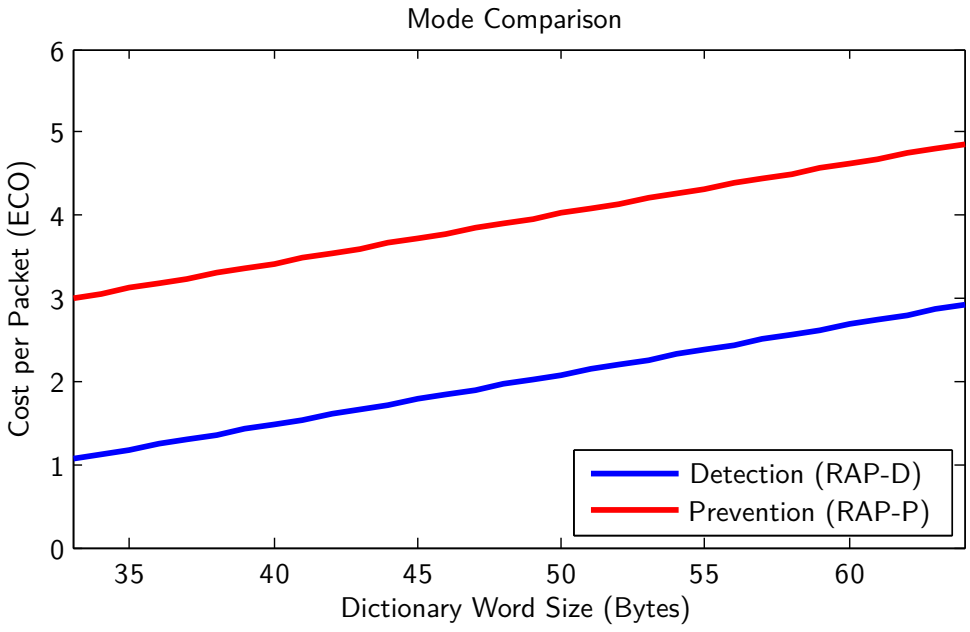


Figure 5.5: The relative cost between RAP-D and RAP-P for the same level of resilience to space exhaustion.

Figure 5.6 investigates the relative cost of the two protocols for different data sizes, by comparing the cost overhead ECO_R for the receiver of the two protocols. Additionally, we consider different dictionary word sizes as requirements for resilience to space exhaustion. The results suggest that increasing the data packet drops the energy cost down for both protocols. The energy overhead of RAP-D can be kept at a minimal level as long as the data size is above the dictionary word size requirement.

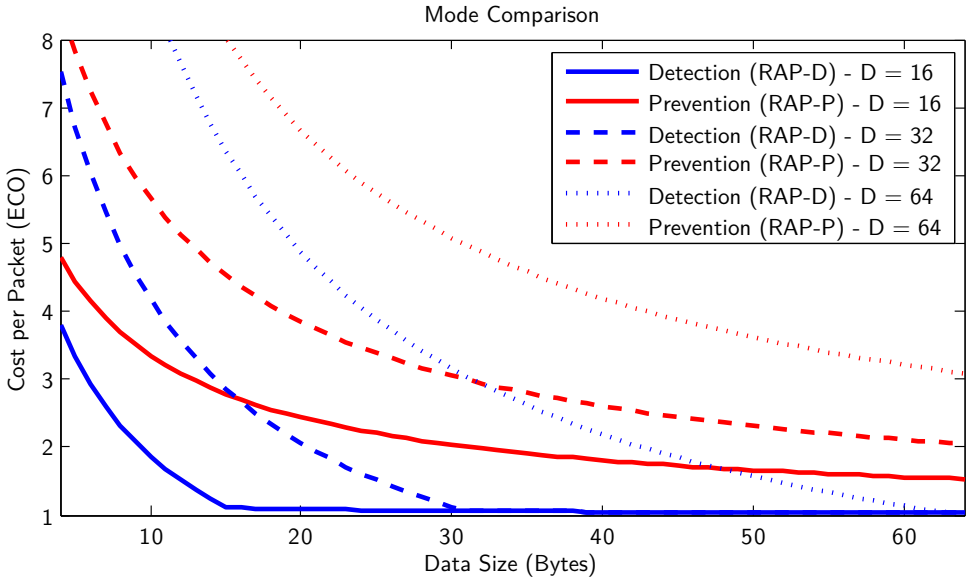


Figure 5.6: The relative cost between RAP-D and RAP-P for different data sizes (L_D) and required levels of resilience to space exhaustion (D).

To conclude on this topic, we focused on securing the class of RI MAC protocols for WSNs against the beacon replay attack. According to the RI paradigm of communication, beacons are used to initiate the communication between two nodes. By collecting and replaying such beacons, an intruder can pretend a fake identity and perform a series of attacks. In particular, we proposed a challenge-response authentication protocol, named RAP, that is able to detect and prevent beacon replay attacks. RAP has two modes of operation. RAP-D is a low-overhead protocol that is able to detect intruders who replay beacons. RAP-P, on the other hand, is a more expensive prevention mechanism. We validated the effectiveness

of RAP against beacon replay attacks using various tools, including OFMC and ProVerif. Furthermore, we have modeled the energy consumption of both protocols and exposed the trade-off between the level of security, measured by the resilience of the scheme to space exhaustion, and the level of energy consumption. Finally, we have shown that the energy consumption of RAP-P is significantly higher than RAP-D, but so are its security guarantees.

Several years ago Microsoft made a big deal about Windows NT getting a C2 security rating. They were much less forthcoming with the fact that this rating only applied if the computer was not attached to a network and had no network card, had its floppy drive epoxied shut, and was running on a Compaq 386. Solaris's C2 rating was just as silly.

Bruce Schneier

6

Adaptive Security

In the previous chapters we have discussed about how we can design protocols that support EH-WSNs. In this chapter we will focus on how it is possible to take direct advantage of different varying levels of energy by providing adaptive security.

The building block of security mechanisms for WSNs are encryption schemes. Independent of the specific application, what normally happens is that the data channel is made confidential and/or authentic through the use of encryption schemes and related modes of operation, as we discussed in the previous chapters. The typical family of algorithms used with sensor nodes are symmetric encryption algorithms since they are considerably less expensive in terms of energy requirements when compared to public key encryption schemes [83]. Different algorithms have different energetic requirements and while some of these are connected to how good and optimized the actual implementation is, a considerable portion is intrin-

sic to the specific algorithm. It is logic to expect that a block cipher with a block-size of 128 bits will require more CPU cycles than an algorithm with a block-size of 64 bits in order to perform similar operations. A similar point can be made for the key-size of an algorithm, a longer key is bound to produce higher energy requirements, despite the fact that it should also increase the complexity of the cryptanalysis and the robustness of the cipher-text.

For this reason, when energy is a big concern, having to commit to a specific algorithm is going to be a sub-optimal decision. In an EH scenario, a specific scheme can be inadequate in different ways: for example it could be too expensive in terms of energy and cause the whole system to delay sending new messages until enough energy has been gathered. Within a network with heterogeneous messages, a given scheme could not meet the security requirements for a particular type of message, while it could be more than enough for a different type.

In order to address this issue we now discuss and propose an adaptive scheme that allows each node to autonomously and independently choose the most suitable algorithm to use for a given link of the network and for a given energy configuration.

6.1 Related Work

Adaptive security is not a brand new concept. The work in [79] uses a similar environment and a similar approach, additionally focusing on priority, but limited to single-hop networks with carrier sense multiple access (CSMA). The authors say that rather than achieving an absolute decrease in energy consumption, they manage to obtain a trade-off among consumed energy, importance of the packets sent and their security.

Another example of adaptive security can be found in [80]. Here optical wireless communications are taken into account. The authors propose to subdivide an encryption system S into n subsystems $S_1, S_2, S_3, \ldots, S_n$ each one representing one encryption parameter such as key size, number of rounds or operation mode. The idea then is to vary those parameters according to the security requirements

or the amount of available energy in the case of battery powered devices.

The work in [31] is closer to static analysis. Three main parameters are used to define the security level of a protocol: the protection level, the probability of an attack and the impact of a successful attack. Concerning the protection level, parameters such as the efficacy of an attack (provided it is successful), the knowledge required to mount it, its cost, the communication overhead and the complexity of the implementation are considered. Similarly, the impact of a successful attack is calculated according to the financial losses during the attack, the cost for recovering from the attack and the losses in reputation suffered by the owners of the system. Finally, the probability of an attack is assumed to be given. These value are composed to obtain a single *security level*. Individual security mechanism are then analyzed and defined in terms of complexity and power consumption. Ultimately, according to the system specifications, the required security capabilities and the provided cost functions, specific security parameters are chosen and the system is run accordingly.

6.2 Protocol Description

We will now introduce and describe the inner working of our scheme on adaptive security. The scheme is extensively relying upon RI MAC protocols which have been introduced in Section 3.4 and further discussed in Chapter 4.

Before doing so, we want to point out that the scheme itself is independent of the specific class of protocol used and can be adapted to work also with sender-initiated MAC protocols. However, given the nature and the specific mechanics of RI protocols, improvements in terms of delay and number of exchanged messages can be achieved. For the design of our scheme we are going to base ourselves upon ODMAC [24], which features a set of interesting capabilities, and is specifically designed to be used with EH-WSNs.

6.2.1 Scheme Description

The scheme we have devised is based upon the idea of adaptivity. The key feature it provides is to allow each node of the network to independently choose the best compromise between security and energy consumption according to different metrics.

A WSN is characterized by nodes producing and exchanging packets. Upon creation, each packet p_i is assigned a security value $h_{p_i} := H(p_i)$, where $H : P \to E \times A$ is a function mapping elements from the set of possible packets P to tuples representing security configurations. This function assesses the criticality of a specific packet. We will abstract from its implementation, but it could be thought as a direct connection between specific parameters of a packet and importance values. For example packets representing aggregate values could be considered more important than single measurements, or potentially harmful control packets (e.g., a message asking to reduce the transmission power) would be rated higher than regular messages.

As described before, the h values are tuples $(e, a) \in E \times A$ where each component directly translates into a specific security configuration of encryption and authorization respectively. Different values are mapped to different algorithms and parameters. This mapping can be decided at design-time of the specific network application. An example can be seen in Table 6.1 where we describe only encryption modes assuming tuples of the form $(e, 0)$. Another possible demonstration can be the default security protocol list of 802.15.4 [36, Table 75], which comprises of encryption, authentication and authenticated encryption.

The protocol relies heavily on the RI paradigm, whenever a receiver node r transmits a beacon, it will include its security capabilities $c^t_{r,\max}$ and $c^t_{r,\min}$, these are respectively the highest h tuple that r can satisfy and the lowest h tuple that r will accept, at time t. A sender node s can then analyze beacons to check if both the destination and the security capabilities of its owner are satisfactory. Assuming a total ordering on the security capabilities, let $\tilde{r}$ be the final recipient for node s (e.g., the base station) and $\Delta(u, v)$ the function that measures the distance in number

$H(x)$	Encryption	Authentication
$(0,0)$	No	No
$(1,0)$	Skipjack (Key size 80 bits, Block size 64 bits)	No
$(2,0)$	Hight (Key size 128 bits, Block size 64 bits)	No
$(3,0)$	AES128 (Key size 128 bits, Block size 128 bits)	No

Table 6.1: An example of the H-Security mapping.

of hops between two nodes u and v, then a beacon b from node r is considered adequate for packet p_i if and only if $\Delta(r,\tilde{r}) < \Delta(s,\tilde{r}) \wedge c^t_{r,\min} \leq h_{p_i} \leq c^t_{r,\max}$ that is, if the distance between p_i and its final destination $\tilde{r}$ decreases by sending p_i to r, and r can satisfy the security requirements of p_i. Note that we use a strict inequality for the distance to account for ODMAC opportunistic forwarding (see Section 4.1.1), where beacons moving a message closer to the final destination of the packet are still considered adequate even if they could be sub-optimal from a routing standpoint.

The pseudo-code for data transmission and reception can be seen in Algorithms 1 and 2. The focal point of these algorithms is the generation of the c values (lines 1.5, 1.6, 2.6 and 2.7). These values are tightly connected to the amount of energy available in a node and to the security policies of the system. The notion of available energy is something constantly varying, especially in EH-WSNs. It may be the case that a node has only enough energy to run in *(No Security, No Security)* mode at the current time. However, that situation might improve after it has been able to scavenge some more energy. On the other hand security policies can impose both static and dynamic values according to the specific type of application. We will now show through some examples how different scenarios can be accommodated by adapting how security values are generated.

Algorithm 1 Adaptive security data transmission.

```
1:  function SEND_DATA(data, dest)
2:      Packet p
3:      Beacon b
4:      p.id ← self.id
5:      p.e ← SET_ENC_CAPABILITIES()
6:      p.a ← SET_AUTH_CAPABILITIES()
7:      p.data ← PACK_DATA(data)
8:      repeat
9:          b ← WAIT_FOR_BEACON()
10:     until Δ(b.id, dest) < Δ(p.id, dest) AND
11:         b.e_min ≤ p.e ≤ b.e_max AND
12:         b.a_min ≤ p.a ≤ b.a_max
13:     TRANSMIT(p, b.id)
14: end function
```

6.2.2 Static Mode

The first scenario that we will describe is defined as *static mode* and can be used to help understand how our scheme works at its core. Senders generate c values for outgoing packets according to the amount of energy available to the node at time of creation, using a lookup-table to match security configurations and energy requirements. A simple definition of this routine can be seen in Algorithm 3, note that here the sender is not taking into account the criticality of the data to choose the security configuration, but rather is using a "best effort" kind of strategy. At the same time, the receiving side of each node is statically assigned security ranges. As a result the system can be seen as a weighted directed graph where an edge from u to v of cost c means that u can communicate to v (is physically in range), but only provided that it uses the security features represented by c. The way to obtain this behavior is to set $c_{\min}$ and $c_{\max}$ to the same value. By doing this a receiver can decide the security class of the packets to accept.

Algorithm 2 Adaptive security data reception.

```
1:  function RECEIVE_DATA()
2:      Packet p
3:      Beacon b
4:     b.id ← self.id
5:      repeat
6:          (b.e_min , b.e_max ) ← SET_ENC_CAPABILITIES()
7:          (b.a_min , b.a_max ) ← SET_AUTH_CAPABILITIES()
8:          TRANSMIT(b)
9:          p ← WAIT_FOR_PACKET()
10:     until p ≠ nil
11:     data ← UNPACK_DATA(p.data)
12:     return data
13: end function
```

Algorithm 3 Encryption capabilities generation in static mode.

```
1: function SET_ENC_CAPABILITIES()
2:     E ← GET_CURRENT_ENERGY()
3:     return encryption_scheme[E]
4: end function
```

6.2.3 Dynamic Mode

The static mode is good for describing how the model works, however its utility is limited. An extension is presented in the *dynamic mode*. WSNs often cover large geographical areas such as forests or fields. It could be the case that treating the whole area as a single zone with some fixed properties is not the best approximation. Imagine an example where a network is deployed in an area covering two different buildings connected by an open space. It is sensible to believe that the nodes inside the buildings will be susceptible to fewer risks compared to the nodes out in the open. For this reason, in dynamic mode, we allow the node at each hop to reconfigure the h value of a packet, increasing it when moving towards a less secure zone, or decreasing it when moving away from such a zone. Here the sender has also to address the importance of the packet as part of the process, making sure that important packets are not under-protected which would result in a security

issue, or that less important packets are not over-protected leading to a waste of energy. Similarly to senders, receivers can adapt their advertised c values depending on the specific area they are in, and the amount of energy currently available. The pseudo-code for the c values generation in this case can be seen in Algorithm 4.

Algorithm 4 Encryption capabilities generation in dynamic mode.

```
1: function SET_ENC_CAPABILITIES()
2:     E ← GET_CURRENT_ENERGY()
3:     Z ← GET_ZONE_ID()
4:     c ← encryption_scheme[E]
5:     if is_low_security(Z) then
6:         c ← c + x
7:     end if
8:     return c
9: end function
```

As a result, a more fine-grained approximation of the area can be achieved, allowing a more aware use of the available energy.

6.2.4 Path Mode

A third way for tailoring our scheme to a specific application is by using *path mode*. Here the idea is to force packets through specific paths by carefully choosing the $c_{\min}$ values advertised by the receivers. Assuming that multiple paths are available to one destination, and that the h value of a packet is related only to its importance, a receiver can dynamically choose to accept different types of packet by adjusting the value of $c_{\min}$. For example imagine that we would like the network in Figure 6.1 to route all the packets containing aggregate measurements through nodes a, b and c, whereas we do not care where single measurements packets are routed. This can be achieved by setting $c_{\min}$ to $(3, 3)$ in a, b, c, and to $(1, 1)$ in the remaining nodes. We also have to make sure that aggregate packets are assigned h values of at least $(3, 3)$ and they will be picked up only by nodes a, b and c as wanted.

Furthermore, the values advertised by receivers can be again dynamically varied according to the situation of the network. For example if nodes a, b and c become

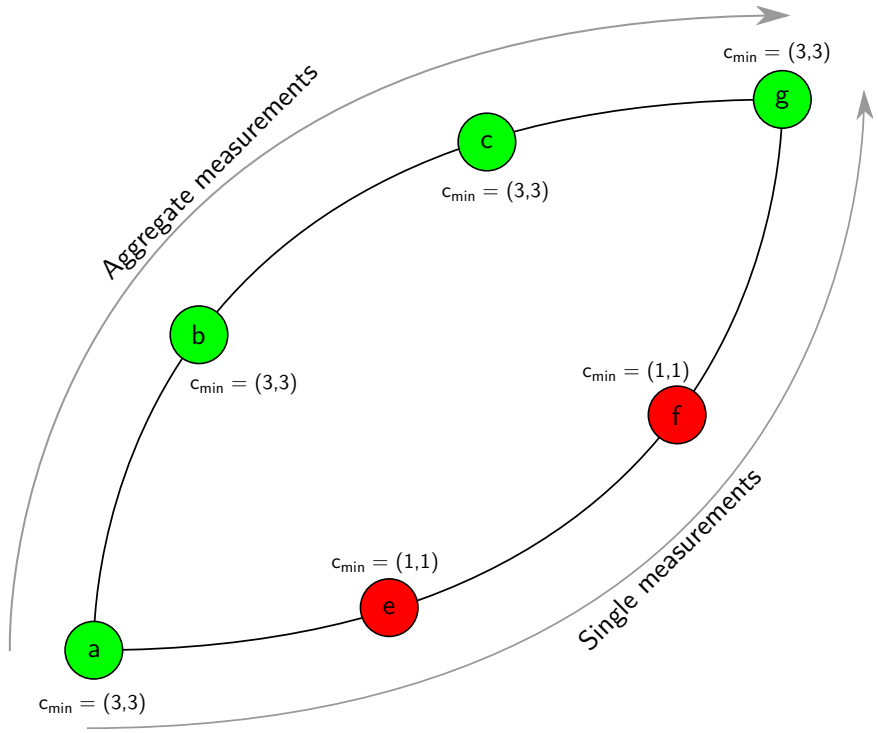


Figure 6.1: Example of how routing can be affected in path mode. The high security packets will be sent through nodes a, b and c, while the low security ones will travel through e and f.

unavailable for a period of time, other nodes can take over their duties by increasing their own c_{min} to accept aggregated packets. Another possibility is to dynamically react to a localized attack (e.g., jamming) by redirecting traffic to a safe area of the network.

6.2.5 Additional Modes

The modes provided above are not "features", but rather they are meant to be convenient names to portray examples and guidelines on how the scheme itself can be adapted to different scenarios and application requirements, and are by no means meant to be exhaustive. Having the possibility to modify the behavior of both senders and receivers allows for considerable flexibility, enabling the design of so-

lutions that are tailored to the problem at hand and hence can guarantee good performances. Imagine an application where delay is a main concern, i.e., packets should arrive from the nodes to the base station as quickly as possible. In this scenario waiting for a beacon advertising the best energy to security ratio costs precious time. It is instead possible to have nodes try to send packets using the first useful beacon, regardless of its security parameters.

At the other side of the spectrum we could have an application where security is the main focus. Here each node could keep track of the security configuration advertised through different beacons, and only use the best seen so far to relay messages, possibly using a weight function that provides diminishing returns according to how old a packet is.

What we have before defined as modes is nothing more than a set a parameters and rules applied to the system and derived by the specific constraints of the application. We feel that it is nigh on impossible to talk about adaptivity and optimization without having a cost function (the specifications of the application) guiding the optimization process. For this reason providing a model that is flexible enough to adapt to different requirements is key in order to obtain good performances.

6.3 Discussion and Considerations

In this section we will provide some points of discussion and some consideration on the scheme in general, including how it would be possible to obtain the different information required by the protocol and some considerations on the security guarantees of our scheme.

6.3.1 Energy Management

Each node has to be aware of the current state of charge (SOC) of its main battery to decide which security values to advertise through beacons. This can be realized with different methods, the most common being Voltage Based Estimation and Coulomb Counting. In the first method it is possible to directly measure

the voltage across the battery and relate this to the actual SOC by means of the discharge characteristics relative to the specific chemistry process used within the cell. For this method to achieve reasonable accuracy, compensation factors for temperature, cell age and discharge rate should be factored in, making it slightly less practical for WSNs. With Coulomb Counting the idea is to consider the battery as a closed system containing a given amount of charge, when full, and subtracting from this value as the battery depletes. In order to measure the actual current drawn from the battery different sensing techniques such as shunt resistors or hall effect sensors can be used.

To correctly asses the cost of a specific configuration it is required to measure the amount of energy needed in order to use it. This can be done empirically, by measuring the state of charge of the battery before and after a large enough number of transmissions and then computing the average in offline experiments.

Once these quantities are known, it is possible for a node to correctly advertise the currently supported configurations. As the state of charge varies over time, less or more configurations will become available and the newly created beacons will reflect the situation adaptively changing the supported features. This is where RI protocols shine, thanks to their core mechanic it is extremely easy to convey information from the sender to the receiver before the actual packet is sent, without having to perform unnecessary communications. The sender can then use these information to decide whether or not the receiver is appropriate.

6.3.2 Security Considerations

Possible attacks to this protocol are closely related to the underlying MAC protocol and the encryption algorithms used. Without touching on the security of the individual algorithms, which is out of the scope of this work, we now analyze what a potential adversary might be able to achieve by manipulating messages within the network and taking advantage of the protocol inner workings. This analysis ties into the correct design of the system and can help define security properties according to the required features. For the adversary model in this section we will

consider a Dolev-Yao attacker [21] which is aware of the protocol and able to eavesdrop, intercept and create new messages using the knowledge accumulated over time.

A first concern is about forging beacons. This would allow the adversary to advertise incorrect security capabilities or malicious routing information in the form or wrong identities. This problem is avoided by ensuring that either an encryption or an authentication layer is always present. The attacker would have to share a key with other nodes in order to be able to communicate fresh messages with them. This argument relies on the secrecy and the strength of the key, which is in line with the attacker model. If for example poor key exchange mechanisms are used within the application and the adversary can get hold of the key, the security of the scheme is obviously defeated.

It is worth to point out that while using either authentication or encryption provides the same upshot (packets can not be forged), the way this is obtained is slightly different. In case of encryption we can say that in order to create a counterfeit beacon, the adversary should produce a key that would allow him to create packets that would be correctly and meaningfully decrypted by recipients nodes, in other words he would have to have a shared key with all the target nodes. The number of such keys depends on the keying scheme used: single-key, probabilistic, group-based or pair-wise, just to name a few.

On the other hand, in the authentication case, the adversary would have to produce a key that can validate the content of the message against a tag appended at end of the message itself, proving that the identity of the owner of the message is legitimate. This is a separate key that can be managed in a completely different way from the other one, for example it could be be a single key which, once compromised, gives the possibility to exchange authentic messages with every other node within the network.

Furthermore, if the system allows nodes to dynamically join the network, it becomes much harder to discover an attacker that tries to disguise himself as a regular node, complies to the protocol long enough to establish a genuine identity and then goes rogue. The result of this is that while both systems guarantee the fresh-

ness of a message, the decision of which is better suited to deal with the problem, as often happens, lies in the details like the key management scheme used.

A second possible course of action for the adversary is to try and spoof or modify received beacons in order to re-transmit them at a later time. This is avoided by using authentication schemes which, by definition, prevent messages to be modified. In other words, by using security suites like our scheme presented in Chapter 4 both integrity and confidentiality are achieved.

Being this and adaptive scheme based upon EH, energy exploitation must be carefully taken into account. While an attacker with physical access to the node could in theory prevent it from recharging and keeping it in a low security state, we believe this is not an effective attack. First of all if an attacker has physical access to a node, energy exploitation is not the main concern, but rather the node could be cloned, reprogrammed or have secret keys extracted from it, all kind of attacks that would cause much greater harm to the whole network. Secondly if the attacker wants to have some kind of distributed effect on the network by changing the current energy parameters, he must do so for a considerable number of nodes, and depending on the actual network size and the kind of energy used to power the nodes, this could be unfeasible.

Finally, one more concern is about replayed beacons. As we have discussed previously, this technique can be used to impersonate another entity, carrying out communications on her behalf and trying to gain some advantage from it. Depending on how this is done, it is possible to force senders to use lower than necessary security settings in order to obtain cryptographic advantage, or to force higher than necessary security settings, thus making nodes use more energy for each message exchange and shortening their active time, possibly causing a denial of service. Other ways of performing this kind of attack are similar in principle but a bit more subtle, for example an adversary could monitor the traffic looking for nodes important to the specific application, like nodes forwarding traffic in a high security path, nodes with a high incoming degree (topology bottlenecks) or nodes performing critical measurements. Once such potential targets have been identified, the attacker can then use a series of replayed messages in order to selectively disrupt the

victims. A solution to this issue can be found in RAP [20], the scheme introduced in Chapter 5 which specifically targets the beacon replay attack. RAP can be used on top of any security mode and can be factored in into the design of the system.

It used to be expensive to make things public and cheap to make them private. Now it's expensive to make things private and cheap to make them public.

Clay Shirky

7

Key Management

Key management is an integral part of security. Confidentiality and authentication strongly rely upon good cryptographic algorithm to encrypt data and compute CMACs. The encryption algorithms themselves require a sound design and a strong key to work properly. All of the protocols and solutions discussed so far make use of these techniques and therefore require good keys. Assuming that the soundness of communication protocols and encryption algorithms holds, we will now focus on how to securely generate and distribute keys in regular and EH WSNs.

Generally, with the term *key management* we identify a series of processes and techniques connected with handling cryptographic keys. Key generation is the first step. In order to securely communicate, two entities (node-node or node-sink) require a so called *shared key*. These keys should be generated in a way that only the intended recipients have access to them. Furthermore, depending on the

protocol in use, having a single shared key may not be enough. As we discussed before, if a security scheme like the one developed for ODMAC provides both confidentiality and authenticity through a single encryption algorithm, different keys should be used for each purpose. This can be achieved in different ways, by generating and sharing additional keys, or by deriving *sub-keys* from a *master key*. If it is allowed for nodes to dynamically join and part the network, these procedures should be repeated to accommodate for the new users. Moreover, if forward and backward security are required, re-keying techniques are needed in order to prevent old nodes to access new messages and new nodes to decrypt previously recorded packets. In addition to this, cryptographic keys have a fixed lifespan, they should not be used to encrypt or authenticate more then a given number of messages. This is usually not a limitation of the key itself, but rather is due to the fact that, depending on the specific scheme, encrypting the same values more than once with the same key could potentially leak unwanted information. In order to prevent this and make each packet unique, additional values are added. However, these values have a fixed length and even by using all the possible combinations there are only so many of them. When the combinations are exhausted, values will repeat. To avoid that, keys should be renewed. Last but not least, if attacks are detected or nodes are compromised, new keys should be distributed once the attack has been dealt with.

7.1 Basic Schemes

We will now discuss some of the typical keying schemes and highlight which are their key advantages and disadvantages.

7.1.1 Single Key

The most simple approach that can be adopted is to use a single key for the system. This has numerous advantages in terms of ease of use. First of all it is possible to *hard-code* the key inside a node at the time of creation. Thanks to this each node has the possibility to interact with every other node of the network without having

to carry out any procedure. This scheme requires an almost negligible amount of memory since only a single value must be stored. Furthermore, it is possible for new nodes to join at any time and start communicating with preexisting ones.

Despite that, the single key scheme falls short in terms of security guarantees. First and foremost it provides a single point of failure. Whenever a node is compromised so is the security of the entire system. With a minimal effort, an attacker is able to effectively become a fully fledged member of the network, able to send authenticated messages, receive messages addressed to other nodes and decrypt all past and future messages.

As a result this scheme is usually only used for demonstration purposes in security protocols due to its ease of implementation, but is should never be used in real deployment.

7.1.2 Pairwise keys

The opposite approach to the single key is to use a different key for each pair of nodes plus the sink. From a security standpoint this scheme offers the best possible security. If an attacker is able to compromise a node and obtain all its keys, only the communications which directly involve this node are compromised. Any other message is secured using a different key to which the attacker has no access. Additionally, it is easy to recover from the loss of a single node, all that is required is to distribute the identity of such node so that every other member of the network can invalidate the specific key used to communicate with it.

Unfortunately, this scheme is extremely costly and it does not meet the scalability requirements of a typical WSN. Given a network with n nodes, the number of necessary keys for the whole system is $n(n-1)/2$. Considering that each node has to maintain a number of keys that increases linearly with the number of nodes $(n-1)$ and that the overall number of keys is quadratic in the number of nodes, this translates to a unsustainable memory consumption for an average node. Assuming a node with 32 KiB of available memory and a cryptographic key of 128 b, the whole memory would be completely filled with keys after only 23 nodes. In

addition to that, for each node added to the network new keys must be generated and distributed.

7.1.3 Random Pre-distribution

Besides being unsustainable from a memory point of view, the pairwise scheme is also an overkill. Assuming that we want to achieve link based security, a message is encrypted/authenticated and sent to the next hop where it is decrypted and checked. The procedure is then repeated for each hop until the final destination is reached. This implies that not every pair of nodes has to share a key, but that one is needed only for links through where messages are actually being transmitted. The idea presented in [23] takes this into account and proposes a randomized scheme where a pre-distribution phase assigns a small set of keys to each node in a way that, with high probability, two nodes connected by a link will share a key. Furthermore, a key generation procedure is used to obtain a key for links that do not have one.

In the pre-distribution phase a large number of keys P (approximately $2^{17} - 2^{20}$) is generated, each node then draws k values from P and uses them as its key-ring. Trusted controller nodes are then used to store a mapping between the identity of a node and the identifier of the keys in its key-ring. Finally each controller node is given the keys shared with each node.

Successfully the shared-key discovery phase takes place. Here each node discovers which key, if any, it shares with its neighbors. This is done by having each node broadcast the identity of the keys in the key-ring or through a challenge-response scheme, depending on whether or not the discovery phase should be public or private. Each pair of nodes that are physically in range one another and share a common key define a *link*.

The third phase called path-key establishment allows nodes physically in range but not sharing a key to obtain one. To do that, pre-existing keys left unused after the shared-key discovery are transmitted to the nodes participating in the path-key establishment.

Once the three phases are terminated each node in range shares a pair-wise key with its neighbors.

The main property of this random scheme is that the probability that the connectivity graph induced by the network is connected can be made arbitrarily large. The authors use a formula derived by Erdős and Réni [22] in the study of random graphs to show how this can be achieved. Let P_c be the desired probability of the connectivity graph being connected and p the probability that there exists a link between two nodes. Then, given a number n of nodes, $G(n,p)$ is a random graph whose probability of being connected is

$$P_c = \lim_{n\to\infty} \Pr\left[G(n,p) \text{ is connected}\right] = e^{e^{-c}} \tag{7.1}$$

$$p = \frac{\ln(n)}{n} + \frac{c}{n} \text{ where c is any real constant.} \tag{7.2}$$

This allows to compute the degree of a node as $d = p(n-1)$ which is the required number of neighbors needed by a node. Furthermore, it is possible to impose connectivity constraints upon the network (required number of neighbors with a shared key) and the key-ring size, and consequently derive the size of the key pool P given a desired probability p' that two nodes share a key. This is obtained from Equation (7.3)

$$p' = 1 - \frac{\left(1-\frac{k}{P}\right)^{2(P-k+1/2)}}{\left(1-\frac{2k}{P}\right)^{(P-2k+1/2)}}. \tag{7.3}$$

We omit the details of how this is derived and redirect the interested reader to the original paper [23]. Instead we present a short numeric example to help clarifying the concept.

Assume a network with $n = 10,000$ nodes and a desired connectivity probability $P_C = 0.99999$. By inverting Equation (7.1) we obtain $c = 11.51$ and Equation (7.2) yields $p = 0.002$. From this we can compute the required degree $d = 20.71$. Hence, if each node has on average d neighbors the network is connected with probability P_c. Furthermore, if we fix the key-ring size to $k = 80$ and

a probability $p' = 0.5$, we can derive the size of the pool from Equation (7.3) to be $P \approx 10,000$.

Finally, the increase of the key-ring size is sub-linear in the size of the key pool. For example if we increase P by a factor of ten, thus making it $P = 100,000$, we have an increase of k of a factor of 3.3 yielding a value of $k = 260$.

7.2 Multipath Key Reinforcement

The scheme described in the previous section is sound and tackles the problem of distributing shared keys within a WSN. However, the keys used therein are simply the keys obtained after the shared-key discovery or the path-key establishment phases. Anyway, all the keys are drawn from a fixed pool and, in order to achieve greater probability of two nodes to share a key while maintaining the size of the key-ring manageable for the memory size of a node, the pool should be kept as small as possible. In contrast, with a small pool there is the concrete possibility that the same key is used on more than one link, therefore if an attacker compromises a node not only all the links that directly involve the node will be compromised, but also any other link that uses one of the keys found in that node. To address this, Chan et al. [9] present a multipath reinforcement scheme whose goal is to strengthen the security by allowing each pair of nodes to use a unique random key. This task can not be solved trivially by generating a sub-key from the already shared key because an attacker that has been recording the key setup messages prior to capturing a node could now decrypt those messages and obtain the new key and therefore access all the messages encrypted with it.

The proposed scheme takes advantage of disjoint paths. Assuming that two nodes u and v want generate a new key from the existing key k_{shared}, then u chooses j different disjoint paths connecting u to v that were setup during the key distribution phase. For each one of these j paths, u, generates a random value x with the same length of the key and sends them to v through the different j paths. After

receiving j many values, u computes the new key $k_{\text{reinforced}}$ as

$$k_{\text{reinforced}} = k_{\text{shared}} \oplus x_1 \oplus x_2 \oplus \cdots \oplus x_j. \tag{7.4}$$

In order to compromise $k_{\text{reinforced}}$, an adversary has to compromise at least one link on all of the j paths. While increasing the number j of paths used decreases the probability of the attacker to succeed, the non immediate trade-off is that the longer the path, the higher the probability of an attacker to compromise at least one link. Moreover, computing the disjoint paths is computationally intensive. To solve this, the scheme uses paths of two hops (three nodes), this makes the discovery procedure less intensive and ensures that they are disjoint by construction. A quick way to find such paths is for u and v to exchange their neighbors table and identify the nodes in common.

Assuming ideal communications, i.e., circular communication range with radius r for both transmission and reception, two nodes separated by a given distance have an expected area of overlap is $0.5865\pi r^2$. Hence, the expected number of reinforcing neighbors (the neighbors common to two nodes trying to run the reinforcement scheme) is given by $0.5865p^2n'$, where p is the probability of two nodes to share a key and n' is the number of neighbors of a node. This can also be expressed in terms of the degree of a node as $0.5865\ d^2/n'$.

We now derive the increase of security achieved by the scheme. Let t be the number of links used to reinforce the key and q_{link} the probability that an adversary will compromise a single node. Then the probability that the adversary will compromise the new key is equal to the probability of compromising either one of the hops in the path, minus the probability of compromising both. By applying this to all the t neighbors and including the original link we have

$$q_{\text{reinforced}} = q_{\text{link}}(2q_{\text{link}} - q_{\text{link}}^2)^t. \tag{7.5}$$

The average overhead of the protocol can be approximated to 10, while the effort required by the attacker to break a reinforced link for a probability $q_{\text{link}} = 0.1$

translates to an increase of 146 times. The scheme experiences diminishing returns, the higher the probability of a node to be compromised, the lower the effort required by the adversary.

7.3 Adaptive Multipath Key Reinforcement

We now take a closer look at how the multipath reinforcement scheme can be applied to EH-WSNs. While the scheme can be run unmodified in this kind of sensor networks it will not take advantage of the core properties of EH. To address that we present a new adaptive scheme that takes into account the available energy of the reinforcement neighbors.

As we have reiterated time and again, contrary to WSNs where the target is to maximize the lifespan of the networks, one of the main goals of EH-WSNs is sustainability, reaching the ENO state. In our scheme this can be achieved by balancing the number of reinforcement links used by the two nodes willing to establish a new key, and the availability of reinforcement neighbors. Both parameters can be adaptively chosen according to the amount of energy available to each node.

Depending on the particular network and to some extent also on the nature of the energy being harvested, different energy situation are likely to be present. We will now describe how the protocol adapts.

7.3.1 Scheme Description

Let us assume that nodes u and v want to run the reinforcement scheme in order to obtain a new key. We define s_u as the required number of reinforcing neighbors for node u and $k_{u,v}$ as the maximum number of neighbors connecting both u and v, that is the size of the intersection of the key-ring of u and v. The value s_u can be chosen in different ways, for example on a message-per-message basis according to the content of the packet or as a global parameter depending on the size of the network and the required maximum probability that an adversary will compromise a link.

If $s_u > k_{u,v}$ then there are not enough available neighbors to run the protocol. One option in this case is to wait for enough nodes to come online. However, given

the unpredictability of EH-WSNs this may never realize, and by the time that new nodes have become available, older ones might have run out of power. In this case if a key must be established in a short period of time we fall back to a centralized scheme where both u and v are assisted by the sink node. This protocol in inspired by the well-known Needham-Schroeder protocol [62] and by [86] where the sink plays the role of the trusted third party. Each node is equipped with a unique key, shared with the BS. The protocol will start with u communicating its intention to establish a key with v to the sink node. The sink will generate a new key $e_{u,v}$ for u and v and a token t_u. These values together with the nodes identities are sent to both u and v, each encrypted with their own BS shared key. After that u will send the token and its own identity to v, encrypting them with $e_{u,v}$. The node v will then decrypt this value and compare the identity of u with the one received from the sink. If the two match, v will encrypt the token under $e_{u,v}$ and send it back to u, thus completing the protocol.

This protocol shifts most of the computational burden towards the BS, however it still requires a significant number of messages to be completed and, most importantly, it is not distributed. Each node relies on the BS and a considerable amount of energy will be spent by the nodes close to it since they will be involved in the majority of the traffic. For this reason we select this protocol only when not enough neighbors are available.

In the opposite case $s_u < k_{u,v}$ then the protocol can be run if the reinforcing neighbors have enough energy available to participate. This value can be advertised by the nodes themselves. An ideal way to do so is by including this information together with the amount of energy available to the node as part of a beacon in an RI protocol. In this way u will receive regular updates with a minimal overhead. When enough neighbors are available u can greedly choose the d_u ones with the highest amount of energy and start the protocol with them.

A node will choose autonomously whether or not it is able to participate to a run of the protocol and advertise that to its neighborhood. One way to achieve that is by setting a threshold on the energy reservoir above which a node is considered to have stored enough energy to participate. The disadvantage of this approach is that

if a node is hovering around the threshold value it might be asked to take place in a run only to find itself without enough energy when the actual ensuing transmission should be performed. To avoid that we define a threshold window $(t_{\text{low}}, t_{\text{high}})$ effectively setting up a comparator with hysteresis. Whenever the energy available to a node is less than t_{low} the node will not participate in the protocol, whereas if the value is above t_{high} the node will consider itself able to participate. If the current available energy falls within the range $(t_{\text{low}}, t_{\text{high}})$ the node will maintain its previous status until one of the other conditions is met. This provides a configurable energy buffer that can be varied according to many parameters such as the typical load of the node, the size of the energy reservoir, the length of a packet, etc. The window can be controlled by varying three parameters. The value of t_{high} will determine how quickly the node will start participating, that is the amount of energy required in order to be considered eligible for the protocol. The value of t_{low} will determine how aggressively the node will participate or how quickly it will transition from the eligible to the ineligible state. The difference $t_{\text{high}} - t_{\text{low}}$ will determine the size of the buffer or how resilient the node is to change its status after a change in its available energy.

7.3.2 Evaluation

In order to evaluate different approaches we have run a series of simulations with different parameters. Three arrangements of thresholds have been used, in each of them the threshold is expressed as a portion of a unity energy reservoir. The first arrangement of thresholds is $t_1 = (0.8, 0.9)$, here the window is small in size and located towards the maximum value. This produces a very conservative behavior where in order to become eligible, a node has to be almost fully charged. The second arrangement is $t_2 = (0.1, 0.2)$ in which the window has the same size but its position is significantly lower. In this case a small amount of energy is required for the node to participate in the scheme and it will keep participating until its reservoir is almost completely depleted. The third arrangement is defined as $t_3 = (0.15, 0.85)$ where the two threshold are set at the midway point of the

previous arrangements, thus making the window size considerably wider. As a result this will provide a more stable behavior where the node will maintain its status for longer and transition from one mode to the other when is either considerably charged or almost out of power.

We evaluated these thresholds in three different network scenarios where we vary s and k. The values we have used are respectively $(2, 3)$, $(2, 4)$ and $(3, 4)$. The simulation continuously runs the reinforcement scheme as the only main task within a node. We let the simulation run for an allotted amount of time while randomly changing the amount of energy harvested by each reinforcing neighbor and keeping track of how many times the protocol was successfully run.

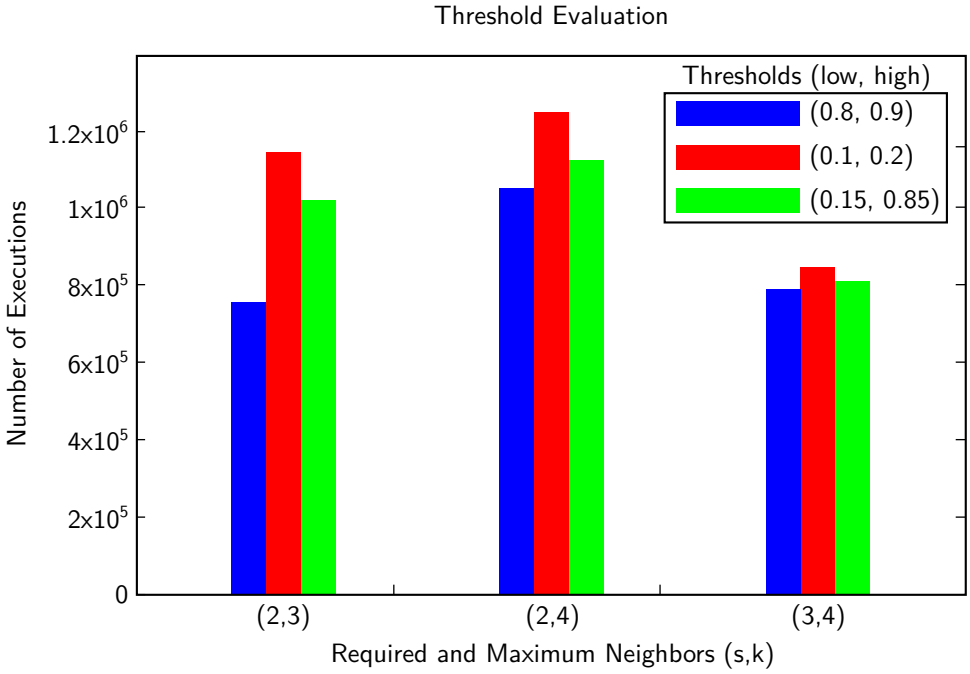


Figure 7.1: Multipath reinforcement for EH-WSNs. Here the thresholds for the hysteresis cycle are statically defined.

As it is possible to see in Figure 7.1, the common trend is that the second configuration, which was the most permissive one, always achieves the highest number of runs, whereas configuration number one, being the most conservative, achieves the lowest amount of protocol executions. Finally, the third configuration presents

a good compromise on the number of runs by keeping the node active most of the time while not letting it run as low in power as the first scheme. This was the expected behavior

The three configurations are a good display of how different parameters can be accommodated by the system. Depending on what is the desired energy status of the nodes, the thresholds can be set accordingly. If the network prioritizes security and therefore wants the scheme to be able to run whenever is required, without worrying of the fact that this could cause some of the nodes involved in the protocol to consume all of their stored energy, then a configuration similar to the first arrangement can be used. On the other hand if we would like a network where energy should be mainly used to exchange messages and perform other tasks, while the key reinforcement should be run only when there is some disposable energy, then the second configuration is the most suited. The third configuration is a high resiliency one and can be used for example when there are relatively short and frequent fluctuation in the availability of the scavenged energy source. If the main energy source were to disappear for a long enough period of time, the first configuration would prevent a node to partake in the protocol almost immediately (as soon as the stored energy started decreasing). The second configuration would instead allow the node to almost run out of energy while still running the protocol. If the third configuration was used in a scenario like this, the node would instead join the protocol only when almost fully charged and therefore be able to sustain a considerable number of executions, but it would not stop right away or exhaust its energy storage when energy becomes unavailable. Once the main source would reestablish itself, the node would simply start charging again without having modified its behavior in the meanwhile.

7.3.3 Sliding Window

One last configuration that we take into consideration is designed to increase adaptability and better suit EH-WSNs. Here we use a sliding threshold window that adapts to the current harvesting rate of a sensor. The window is allowed to shift up

and down, and to shrink or expand depending on the current harvesting rate. The idea in general is to start in a configuration similar to the first scenario presented before with a small window positioned at the top. As we know this is a conservative approach and is a good starting point for when the node first comes online and has not much energy available. As the harvesting rate and the available energy increase, the window will start moving down and increase in size, reaching a configuration that is halfway between arrangements number two and three. If the harvesting rate becomes negative, the window will revert to its previous state by floating back up and reducing its size.

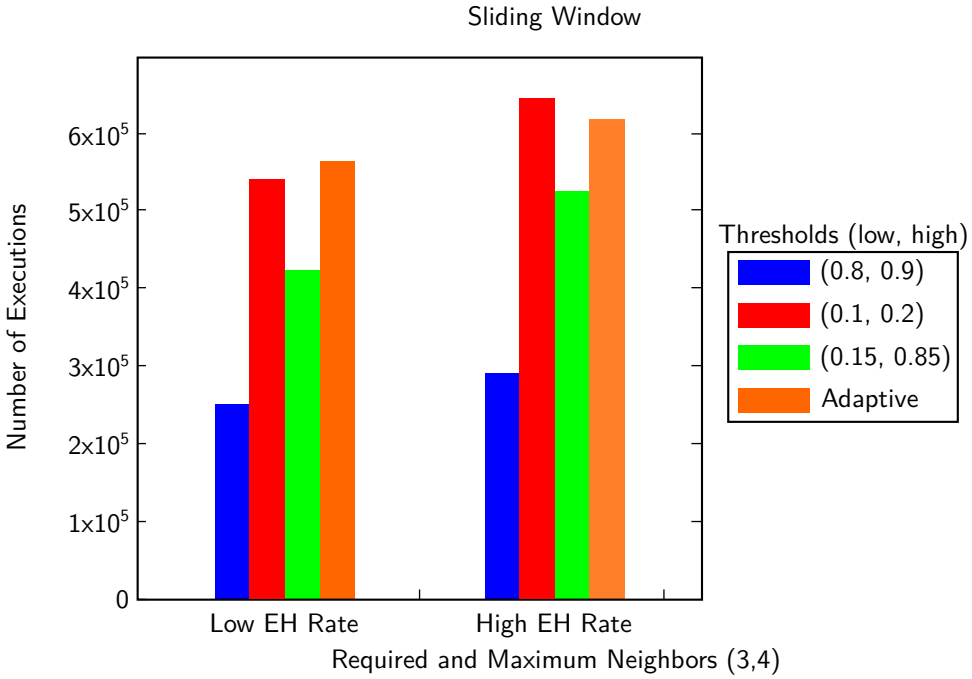


Figure 7.2: Multipath reinforcement for EH-WSNs. Both thresholds for the participation of a node move according to the amount of energy available, making a node more likely to participate if it has abundant energy.

As shown in Figure 7.2 the adaptive configuration performs similarly to the second configurations, however, it will prevent a node to run too low on energy by increasing the minimum threshold when there is not enough energy. Another advantage of this approach is that the nodes that are harvesting more energy will be

the ones that will take part in the protocol more often. Furthermore, this concept remains true in an adaptive way, meaning that if the energy source will change in such a way that some nodes will not harvest as much energy, but others will start harvesting more, then the second group will take over the duties of the first one.

As with the other scheme, the initial level of the thresholds and the expansion-contraction rate are system parameters that can be tuned according to the application and the energy source in use.

It's alive! It's alive!

Henry Frankenstein (Colin Clive)

8 Implementation

In this chapter we will put together some of the different pieces introduced in the previous chapters and we will present and discuss the implementation work done on ODMAC. We will describe the platform of reference and show some practical measurements.

8.1 Hardware Overview

The core functionalities of ODMAC have been implemented on real nodes. The platform we have used it the eZ430-RF2500 Wireless Development Tool produced by Texas Instruments [40] which is a commercially available development platform and supports *C* language.

The board features a Texas Instruments MSP430F2274 ultra low-power MCU [37] (Figure 8.1) capable of running at up to 16MHz. The controlling unit has a con-

sumption of 270µA@1MHz, 2.2V and a standby consumption as low as 0.1µA. It uses a 16-bit architecture and features two internal 16-bit timers, a 10-bit ADC, 32 KiB of flash memory and 1 KiB of RAM. Packaged inside the ADC and powered by a constant current source connected to the internal reference there is a positive temperature coefficient temperature sensor which allows the MCU to access the temperature of the die and, to some extent, approximate the external temperature.

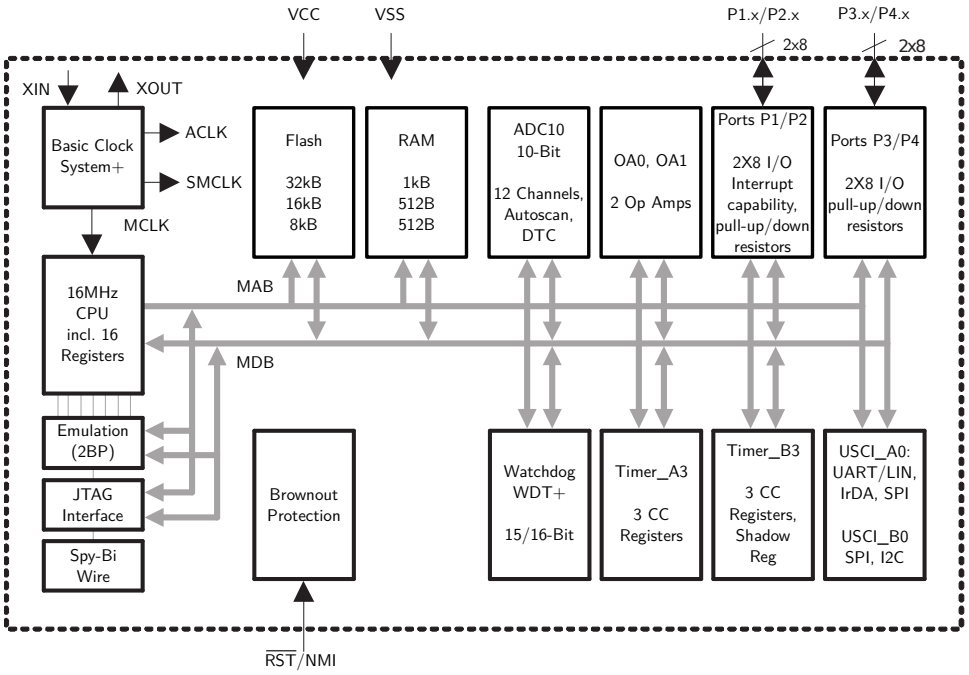


Figure 8.1: Block diagram of the MSP430 MCU [37].

Several operating modes are available for this chip, one active mode (AM) and four low-power modes, each of them providing a different set of functionalities and a smaller number of active peripherals as shown in Table 8.1. When in any of these modes, the device can be awaken by an interrupt event, handle the associated routine and return to the same power mode previously selected.

Three different clock signals are available, the main clock (MCLK) is used for the system CPU, the sub-main clock (SMCLK) used by the peripheral modules and the auxiliary clock (ACLK) provided for the user. Each clock can be driven by

Mode	Description	Consumption (μA) @1 MHz, 3 V
Mode	All clocks are active	300
LPM1	CPU disabled	55
LPM2	CPU and MCLK disabled	17
LPM3	CPU, MCLK, DCO disabled	0.9
LPM4	All clocks are disabled	0.1

Table 8.1: *MSP*430 low-power modes, each mode reduces the available peripherals and therefore also the energy consumption of the chip [38].

one of the possible sources that include a 32, 768 MHz watch crystal, the digitally-controlled oscillator (DCO) (1 – 16 MHz) and the very-low-power low-frequency oscillator (VLO) ($\sim$ 12 kHz).

The various clock sources can be also used to drive the two timers (Timer A and Timer B) to allow different timings of interrupt generation.

The board also features a CC2500 Chipcon 2.4 GHz low-power RF transceiver [39] (Figure 8.2). This has a programmable data rate from 1.2 to 500 kBd and typical current consumption of 17.0 mA for reception above sensitivity limit, 21.2 mA for transmission @0 dBm and 900 nA in power-down mode.

The advantage of this board is that it accepts different power adapters. A USB dongle allows the node to connect to a computer and behave as a BS, while a battery adapter allows for cordless operation. Furthermore it is possible to connect third-party power supplies to provide energy in different ways. Conveniently for us Cymbet produces a series of compatible EH products including different evaluation kits of which we have used two in particular.

The first board is the CBC-EVAL-10 demonstration kit [13] (Figure 8.3) which is an EH solution designed to connect to photovoltaic cells whereas the other broad, called CBC-EVAL-09 [12] (Figure 8.4) accepts multiple types of energy at its input.

As it is possible to see in the block diagram in Figure 8.5, the board accepts: low voltage (below 4.06 V) DC input, low voltage (below 4.06 V) AC input, high voltage (between 4.06 V and 20 V) DC input and high voltage (between 4.06 V

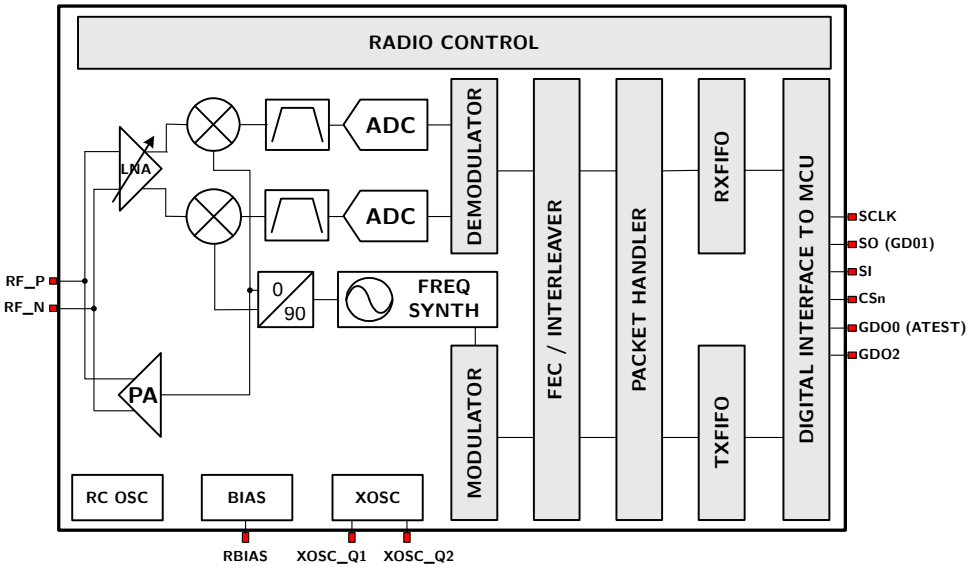


Figure 8.2: Block diagram of the CC2500 RF transceiver [39].

and 20 V) AC input.

The input selection constitutes the main difference between the two boards. Both of them are in fact equipped with the same *EnerChip* CBC51100 module which comprises of two 50 µAh solid-state batteries for a total capacity of 100 µAh and can optionally accommodate external rechargeable batteries.

8.2 Implementation of ODMAC

We have developed a proof of concept implementation of ODMAC, and we will now discuss some of the details.

8.2.1 Core Functionality

One of the most important features to obtain was to allow the system to switch between active and sleeping states, in other words to implements DCs. As we have introduced in Chapter 4, ODMAC has two different and independent DCs, one is used for producing beacons and forwarding messages, while the other is used

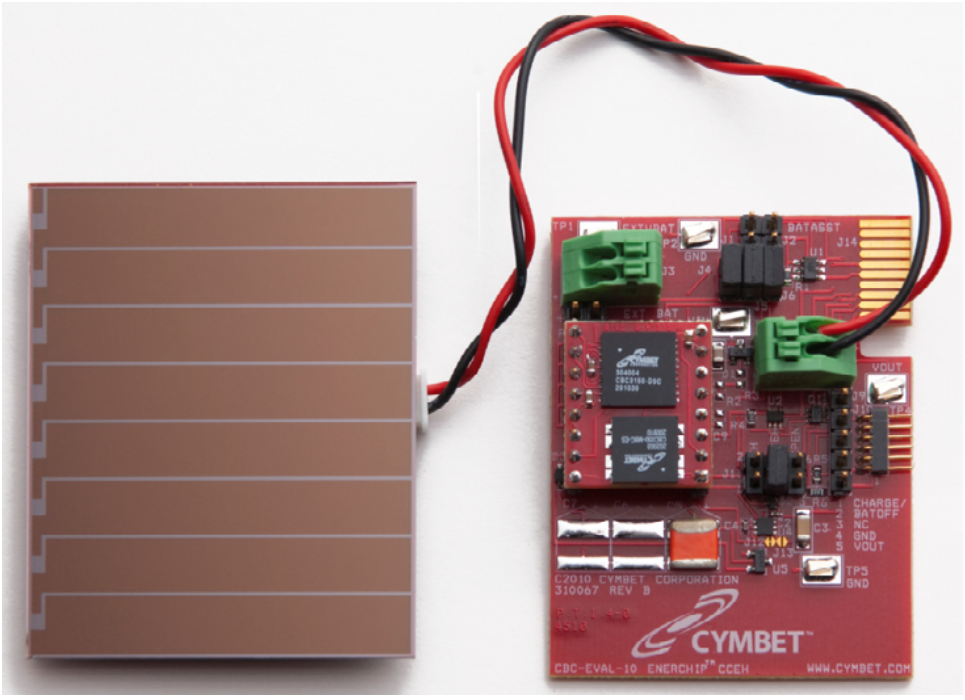


Figure 8.3: The CBC-EVAL10 board.

to fulfill the sensing tasks of the node. To produce the desired effect we have programmed nodes to continuously run in LPM (specifically LPM_3), and awake them by means of interrupts. The chosen LPM allows the ACLKs to remain active, this clock signal is used to source Timer A, allowing it to trigger at regular intervals. In order to implement the two DCs we have decided to use a fundamental time quantum T and to derive the DCs as its multiples. This allows us to have independent periods $T_{\mathrm{B}} = t_{\mathrm{B}}T$ and $T_{\mathrm{S}} = t_{\mathrm{S}}T$ for beaconing and sensing respectively.

Every T cycles the main period interrupt is fired and, by keeping track of how many periods have elapsed, the system reacts accordingly. After having performed the required operations for the current period, if any, the node goes back into the designated LPM and the procedure starts over. One additional factor is that the duration of the quantum is not constant, instead at each iteration it is randomized by a number of cycles in the range $[-2^{r-1}, 2^{r-1}]$ where r is a system parameter that, in our implementation provides a variation of roughly 12 %. This is done to avoid

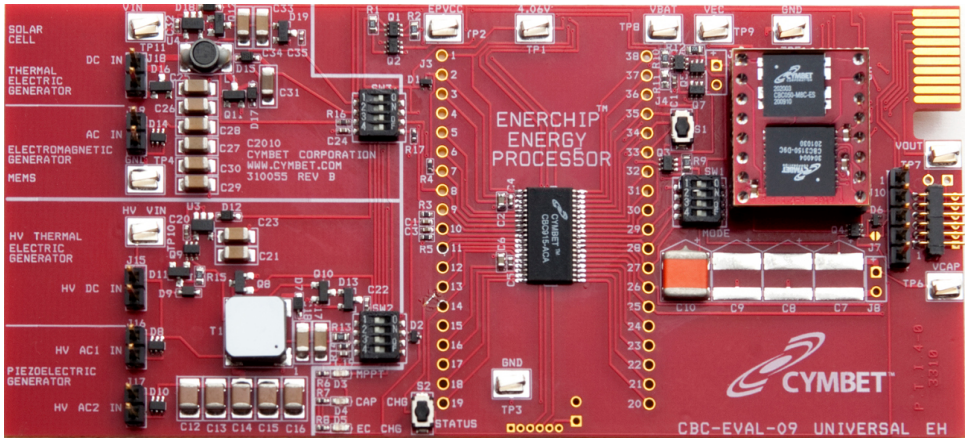


Figure 8.4: The CBC-EVAL09 board.

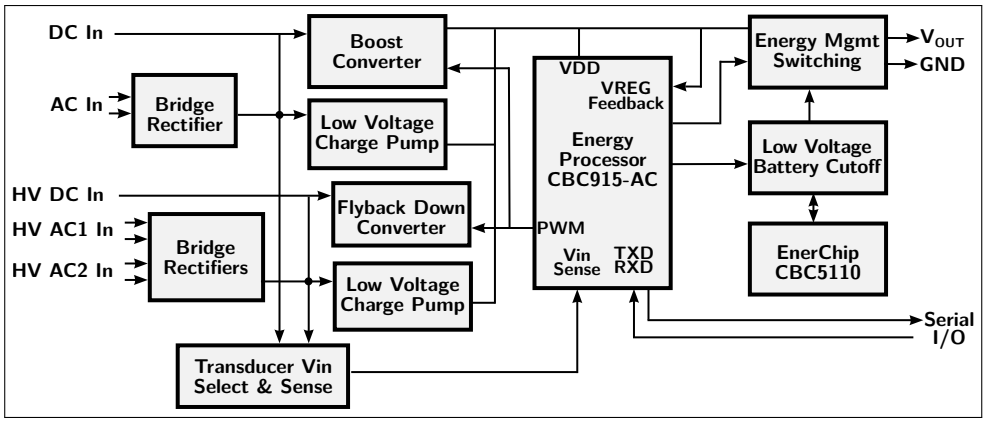


Figure 8.5: Block diagram of the CBC-EVAL09 board [12].

unfortunate synchronization situations where two (or more) neighboring nodes would wake-up at the same time and remain synchronized until one of them would run out of energy. This is an undesirable situation if they both want to receive from or transmit to a third node. The pseudo-code of the main loop of ODMAC is shown in Algorithm 5. There we can see the main system parameters described above used to initialize the fundamental time quantum, the baconing period, the sensing period and the randomization value.

The two main tasks carried out during the activity of the node are the fundamental routines *send* and *receive*. Within them, the RI paradigm is implemented.

Algorithm 5 Main loop of ODMAC.

```
1:  function SETUP(t, t_B, t_S, r)
2:      T ← t
3:      T_B ← c_B ← t_B
4:      T_S ← c_S ← t_S
5:      R ← r
6:      START_TIMER(T)
7:      SET_MODE(LPM3)
8:  end function
9:
10: function TIMER_INTERRUPT
11:     c_B ← c_B − 1
12:     c_S ← c_S − 1
13:     if c_B = 0 then
14:         RECEIVE()
15:         c_B ← T_B
16:     end if
17:     if c_S = 0 then
18:         SEND()
19:         c_S ← T_S
20:     end if
21:     T ← RANDOMIZE(R)
22:     START_TIMER(T)
23:     SET_MODE(LPM3)
24: end function
```

The send routine (Algorithm 6) in this implementation is also an abstraction for generating data. Overall the following operations are performed: first a packet is formatted, this can happen in two different ways, the packet is generated locally by accessing the on-board sensor and extracting the data, or the data comes from a packet received from another node and must be forwarded. When the packet is ready for transmission the node will turn on its radio and wait for an appropriate beacon. When one is received the packet is transmitted and the node returns to a sleeping state.

On the other hand, the receive routine (Algorithm 7), is dedicated to forward-

ing messages coming from another node. When its handler is invoked, the node will create and transmit a beacon containing the relevant information. The node will then wait for messages for a given amount of time. If no message is received the node will simply go back to its sleeping state. In case the beacon is instead answered the received packets is processed and fed to the send routine which will forward it to the next hop.

The packet format for regular unsecured packets can be seen in Figure 8.6.

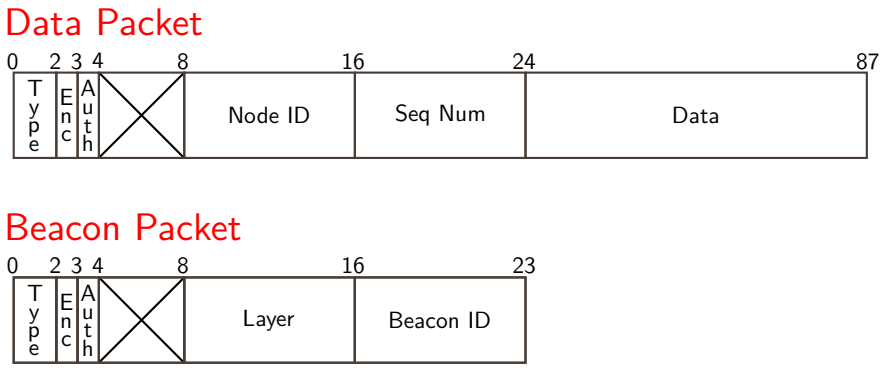


Figure 8.6: Unsecured packet format for ODMAC. No authentication mechanism is available and data is sent in the clear.

8.2.2 Security implementation

The security suite frameworks previously described in Chapter 4 have also been implemented. This extra functionality requires some modification to the transmission routines, but it is generally transparent to the user which only has to set the desired mode. When a packet is created and about to be sent, it is first encrypted (if required) and then authenticated (again, if required). The well-known advantage of applying those transformations in this order is that integrity of the cipher-text is provided and, as a consequence, also integrity of the plain-text. Furthermore, it is not possible to maliciously modify the cipher-text so that it will decrypt to some other (meaningful) plain-text, and finally assuming that the output of the encryption will appear to be random, so will the result of the CMAC, preventing structural information to leak through.

When a secured message is received the opposite procedure is performed. First of all the CMAC is verified, if there is a mismatch the message will be discarded and the node informed with a corresponding error. This prevents malformed messages to have an impact on the whole network by being forwarded to other nodes, causing them to spend unnecessary energy in the process. Once the message is authenticated it will be decrypted and the normal behavior of the node will continue. The packet format for the highest security mode (authentication and encryption) is shown in Figure 8.7.

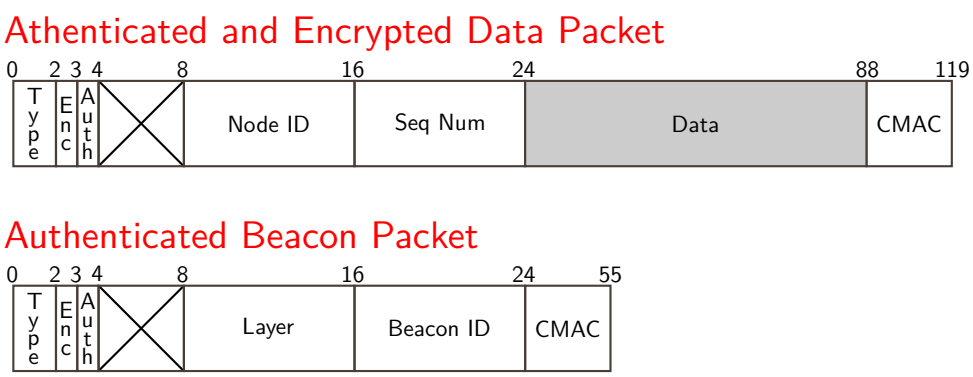


Figure 8.7: Fully secured packet format for ODMAC. An authentication code is added at the end of both types of message, and the data field is also encrypted.

8.2.3 Routing Implementation

The distributed LARs scheme is also part of the implementation. The function $layer(u)$ is used to compute the distance between node u and the BS in number of hops. The initial setup phase consists of setting the sink node with a layer value of zero. Upon receiving beacons each node will compute its own layer as a plus one increment to the minimum layer value received so far. After that, a beacon is defined as appropriate, and can therefore be used to exchange a message, if and only if its layer is less than the layer advertised within the beacon. Since nodes can only decrease their layer value and never increase it, they would not react correctly to the addition of new nodes. To avoid that, these values are reset after a fixed

amount of time if no appropriate beacons have been received.

Algorithm 6 Data transmission in ODMAC.

```
1:  function SEND()
2:      Packet p
3:      Beacon b
4:      p.id ← self.id
5:      p.layer ← self.layer
6:      if p.fwd_data = false then
7:          p.data ← READ_FROM_SENSOR()
8:          p.fwd_data = true
9:      end if
10:     if p.enc = true then
11:         p.data ← ENCRYPT(p.data)
12:     end if
13:     if p.auth = true then
14:         p ← AUTHENTICATE(p)
15:     end if
16:     repeat
17:         b ← WAIT_FOR_BEACON()
18:     until IS_APPROPRIATE(b) = true OR TIMEOUT() = true
19:     if TIMEOUT = false then
20:         TRANSMIT(p)
21:     end if
22: end function
```

8.2.4 Experimental Results

We have conducted some experiments with our proof of concept implementation imagining different scenarios. We consider a delay-tolerant application focused towards throughput, such as a long-term data monitoring for off-line analysis. Here we assume that quickly receiving data packets is not crucial since the analysis will be performed at a later time. What is important instead is the sheer number of packets successfully received. The more the date, the better the analysis.

To experiment in this scenario we focus on a single link, hence we use two regular nodes: a sender and a receiver. The sender will wake up according to its DC,

Algorithm 7 Data reception in ODMAC.

```
1:  function RECEIVE()
2:      Packet p
3:      Beacon b
4:      b.id ← self.id
5:      b.layer ← self.layer
6:      repeat
7:          TRANSMIT(b)
8:          p ← WAIT_FOR_PACKET()
9:      until p ≠ nil OR TIMEOUT() = true
10:     if TIMEOUT = true then
11:         return TIMEOUT_ERR
12:     end if
13:     if p.auth = true then
14:         if CHECK_TAG(p) = false then
15:             return TAG_ERR
16:         end if
17:     end if
18:     if p.enc = true then
19:         p.data ← DECRYPT(p.data)
20:     end if
21:     return p.data
22: end function
```

perform a reading from the sensor, encrypt and authenticate the packet and transmit it to the receiver node by waiting for a beacon and then actually sending the packet. In Figure 8.8 the current consumption of a node is shown, and it is possible to see how each of the aforementioned phases are clearly reflected in the plot. The majority of the current is used by the radio in listening mode while waiting for a beacon. This was the expected behavior.

To test our system we have programmed two different kind of receivers node with different DC values. An high DC receiver, which is set to approximately 33 ms of sleeping time between two consecutive beacons, and a low DC receiver where the same value is set to 66 ms. With this configuration we found that the average activity duration for the transmitter node was $\mu_{\text{high}} = 43$ ms with standard devia-

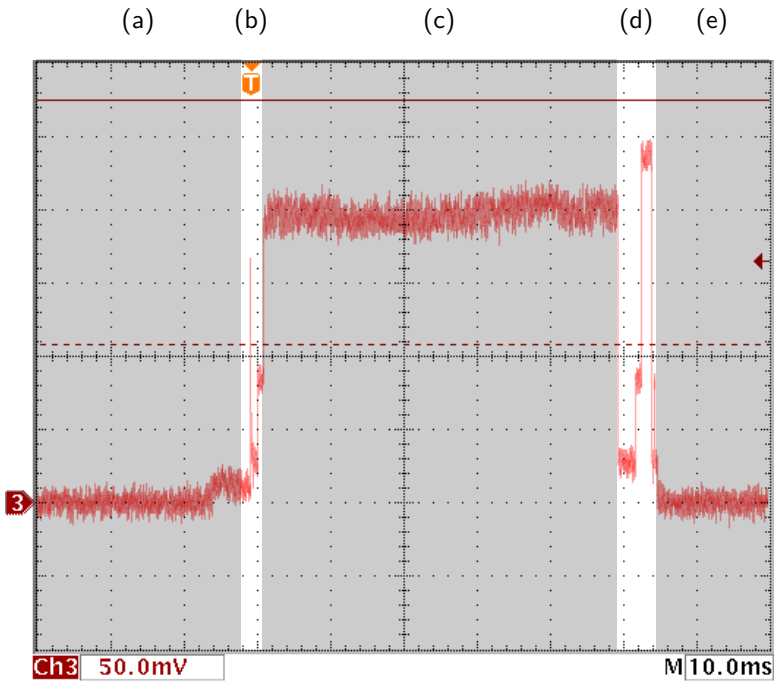


Figure 8.8: Current consumption of a transmitter node during a typical active period. The waveform was obtained by measuring the voltage across a 10 Ω shunt resistor in series with the power supply. It is possible to divide the waveform into four different sections, each one corresponding to a specific phase of the active period. Specifically, from left to right we have: sleep end, packet generation, waiting for a beacon, packet transmission and sleep start.

tion of $\sigma_{\text{high}} = 11$ ms for the high DC receiver, and $\mu_{\text{low}} = 61$ ms with a standard deviation of $\sigma_{\text{low}} = 23$ ms for the low DC receiver.

We also consider an EH case. The receiver node in this experiment was powered through one of the EH boards introduced before, specifically the CBC-EVAL-09. The intended purpose of this board is to provide power to the node when it requires it, mainly during MCU activity and data transmission. To do so, the energy stored within the two solid state batteries, which in turns are replenished by a photovoltaic cell, is used to charge-up a 1000 µF output capacitor that will then provide power to the node itself. While this technique works well for low and sustained current requests, it does not support current spikes which would deplete

the capacitor and shutdown the node. Furthermore, a safety mechanism kicks in and the load is disconnected until the capacitor is fully charged. This is behavior is independent of the SOC of the batteries which simply do not have enough time to recharge the capacitor if this is overdrawn. As described in an application note [11] the minimum time for the batteries to replenish the capacitor is 10 s.

We have empirically established that active periods of around 150 ms are tolerated by the system and do not cause a reset. To address the issue in a definitive manner, instead of replacing the capacitor with a larger one, we decided to use a feedback loop. We measured and conditioned the voltage across the output capacitor, and fed it back to the of the internal ADC of the MCU. By doing so the node is aware of whether or not there is enough energy to perform a message exchange. As a consequence of this, not all the active periods are used, but only the ones where the capacitor is sufficiently charged. We have set the time quantum to one second and the transmission multiplier $t_S = 10$ in order to have an overall transmission period $T_S = 10$ s. We consider a period usable whenever the nominal value across the capacitor is above 3.3 V.

The result of this operation is to have a variable DC that automatically adapts to the current amount of energy and to the harvesting rate. In Figure 8.9 it is possible to see the voltage across the capacitor and its peculiar charge/discharge cycle. Different factors come into play here. The "depth" of the discharge is affected by how long the radio is turned on, and by the current charge value, the recovery time depends by the previous charge value (how much of the energy stored within the capacitor was used) and the current harvesting rate. Finally the number of discharges is associated with the usable active periods and depends an all this factors, the less energy used in the previous period and the shorter the recovery time, the more likely it is that the next period will be usable.

In order to try different harvesting rates we shone a controllable light source towards the photovoltaic panel connected to the harvesting board. By varying the distance we were able to vary the light intensity and therefore the harvesting rate of the node. We then conducted subsequent experiments of 30 minutes each. Considering that the amount of energy inside the capacitor is sufficient for two to three

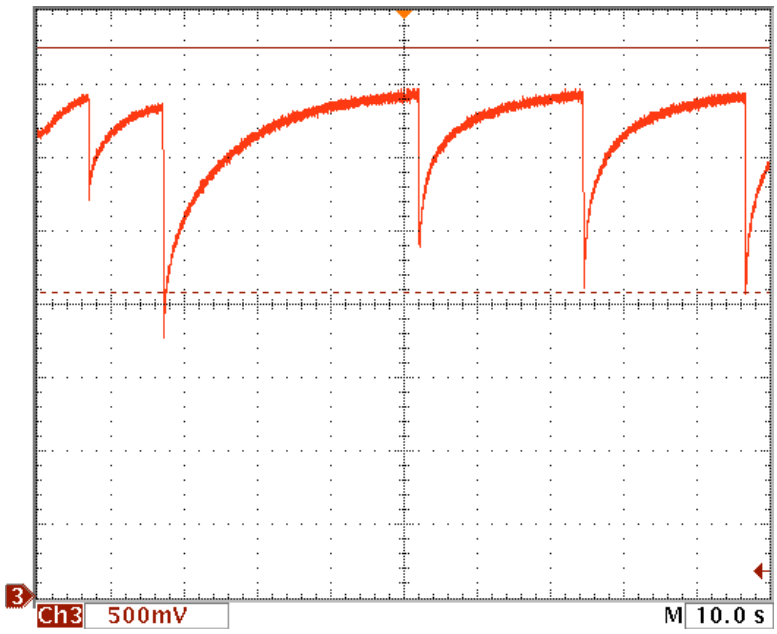


Figure 8.9: Voltage across the output capacitor, each charge/discharge cycle represents an active period. The likelihood of a period to be considered usable depends on many factors such as the amount of energy left inside the capacitor during the previous period and the harvesting rate.

packet exchanges in the best possible conditions, the continuous operation of the node for the whole duration of the experiments confirms its sustainability.

Another interesting fact is that more energy available allows the system to recharge more consistently and therefore more periods will be usable to perform message exchanges. As a consequence of this, more packets will be sent and the throughput will increase, as initially desired for this application. The ENO-Max state of the system is given by the time quantum. Assuming that enough energy is available no more than one packet per ten seconds (the active period frequency) can be sent.

Using the current consumption of a node and the voltage across the solar panel, we estimated the power consumption of a node and plotted it against the achieved throughput as shown in Figure 8.10. As it is possible to see the throughput is proportionally dependent to the input power and furthermore in the high-DC receiver case, where a beacon is sent every 400 cycles, this values is also capped by

the hard limit of 6.0 packet/min (1.0 packet/10 s).

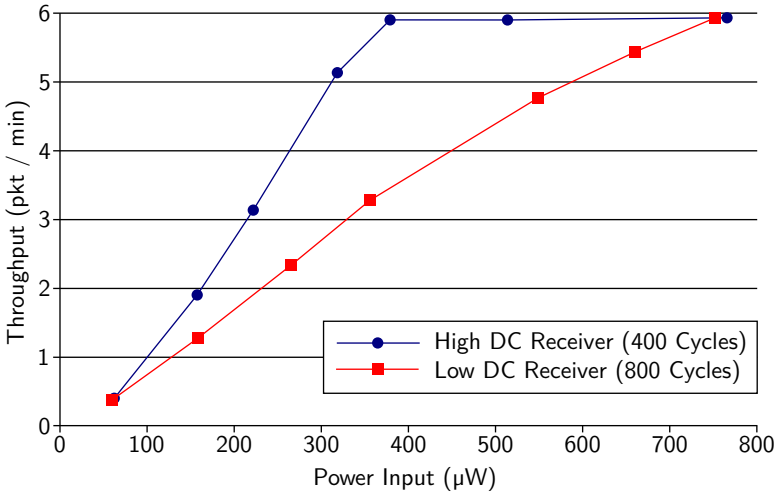


Figure 8.10: Performance of different nodes at different power levels. When the available power is more than the amount used by the normal operation of a node, the throughput reaches the software limit.

A second scenario that we have devised focuses on a security-centric application. We consider a situation where two kind of packets are available to a sender node: low security and high security ones. Low security packets will be sent unencrypted, whereas high security ones must be encrypted and authenticated. Whenever possible we give precedence to high security packets, meaning that if enough energy is available an encrypted and authenticated message will be sent. An unsecured message will be sent otherwise. We used the same configuration with a single link used in the previous experiment and, for low security packets we kept the same threshold voltage to determine whether or not an active period is usable. However, we used a different threshold increased by 12.5%, to establish whether a period was usable for high security packets.

The normal behavior of the sender node would then be the following: upon waking up it would check the capacitor value, if the value is above the higher threshold the node would send a high security packet, if the value is between the high and the low threshold the node would send a low security packet and if the value is below the low threshold the node would go back to sleep.

We first run a control experiment only sending low security packets, thus disabling the security overhead. Afterwards, we have run the full experiment with both thresholds. Both experiments lasted for 90 minutes and are summarized in Figure 8.11. In the control case we were able to send 525 packets, averaging 0.097 packet/s (0.97 packet/10 s). This value is very close to the theoretical limit of 1.00 packet/10 s. For the full-blown experiment we managed to transmit 486 packets over the entire duration, 397 of which were high security (encrypted and authenticated), while the remaining 89 were low security. As a result we achieved a cumulative average of 0.090 packet/s (0.90 packet/10 s) and an average rate of 0.073 packet/s (0.73 packet/10 s) for high security packets only.

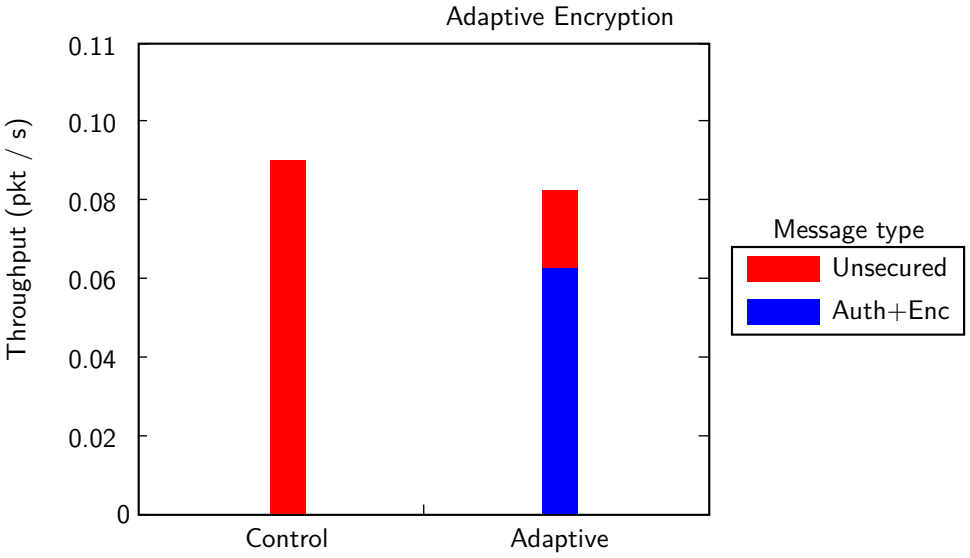


Figure 8.11: The adaptive, security-enabled scheme has tighter energy requirements but shows a decrease in packets sent of only 7.42%.

Despite having added more functionalities and tightened the energy requirement by increasing the transmission threshold, we were able to maintain extremely similar performance. This proves that the impact of the security suite is negligible.

8.3 Test-bed Deployment

Finally, in order to provide a real environment, we have deployed a small temperature monitoring network consisting of ten nodes, with one of them operating as sink (Node 0) and the rest as sensor nodes. Node 1 is powered by the solar energy harvester board introduced before, while the other sensor nodes are powered by batteries. The nodes are forming a multi-hop topology, and are positioned as shown in Figure 8.12. The battery-powered nodes are generating one data packet every 15 minutes and one beacon every 2 seconds. Node 1 reports once every 30 minutes and generates a beacon once every minute. Note that this configuration is sustainable, i.e., the amount of power harvested (approximately 4.9 μW) allows for a continuous operation. Authentication and encryption are activated for all transmitted data packets. Additionally, all beacons are authenticated.

Due to the lack of acknowledgments and retransmissions, we experienced significant packet loss for the nodes that were deployed far away from the sink, i.e., Nodes 4–9. In the case of the nodes deployed close to the sink, that is nodes 1–3, approximately all the packets were received. Additionally, we observed that Node 4 forwarded most of the traffic for Nodes 5–9. This phenomenon can be explained by its physical position, which keeps the beacon and data packet error rate significantly low.

It is important to mention that this particular setup was still in the early stages of the development of the firmware and is presented mainly as a proof of concept for a sustainable operation.

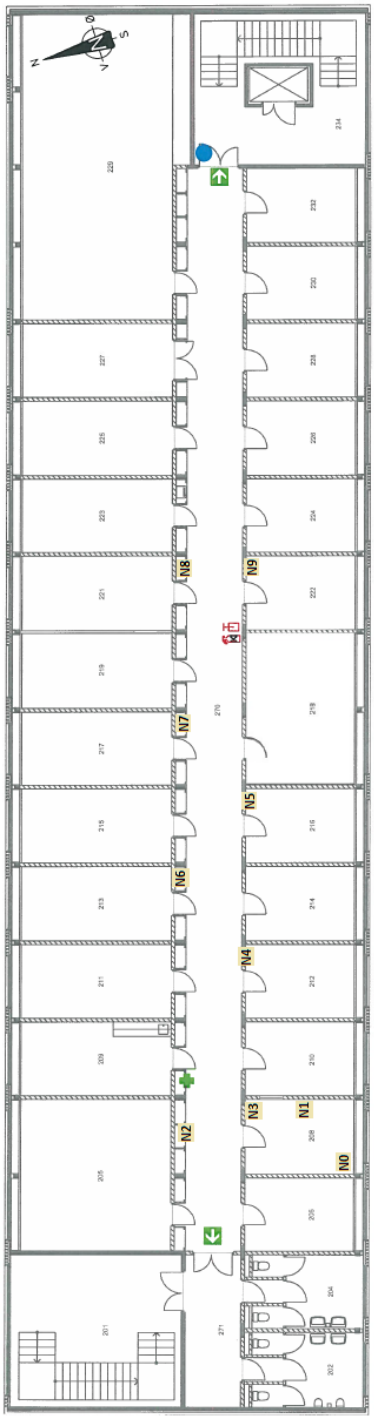


Figure 8.12: The topology of the deployed test-bed resulting in a multi-hop network.

A conclusion is simply the place where you got tired of thinking.

Dan Chaon

9 Conclusion

This study set out to explore and better understand security in EH-WSNs. Security is an all-important task in any branch of computer science; it has been and it continues to be a hot research topic both in general and for WSNs in particular. EH on the other hand is a very new an promising scenario which has gained more and more popularity over the past few years, especially after being applied to the field of WSNs. This study sought to combine these two aspects by analyzing how they can interact and take advantage of one another. We wanted to understand how a non-monotonic change in energy could affect the security of a sensor network and how the highly inconsistent and ever-changing energy configurations of single nodes would influence the network as a whole.

To do this we started off by looking at the big picture, identifying and defining attackers that are specific to this particular domain, through a formal approach. We have described new possible actions that attackers can perform to take advantage

of EH and its peculiar characteristics. To better understand this, we have organized different attackers in a taxonomy and composed a list of attacks matching our taxonomy. One of the first results obtained from this was that the commonly used Dolev-Yao attacker model is indeed an over-approximation of many aspects of what an attacker can do in both a regular WSN and an EH-WSN, and yet this model is not considering other aspects of the same scenarios. We therefore highlighted the need for a cyber-physical attacker which could better take into account the unique challenges of sensor networks.

We then concentrated on securing EH-WSNs specifically focusing on one of their building blocks, MAC protocols. They constitute the hart of most sensor networks by controlling how and when the wireless channel is accessed and how the radio is used. This is where most of the energy budget of a node is used, and in our minds it had to be first point to tackle. During this process we discovered how the RI paradigm was particularly well-suited for this application and therefore turned our attention towards them. This family of protocols use small packets called beacons to manifest the intention of a node to receive data. This was a good match for our needs, but we realized that none of the RI protocols already available would allow EH-WSNs to achieve the best performance so we started designing what would become ODMAC.

We introduced this protocol in Chapter 4. ODMAC is an RI protocol that is designed to address the issues typical of EH-WSNs. Specifically, it allows each single node of the network to independently select its own DC according to the current amount of energy available. We set out to secure ODMAC and designed a small, inexpensive and efficient protection mechanism that can provide both confidentiality and integrity. By operating at the link layer it also allows to tackle security as soon as possible making sure that malformed or maliciously forged packets are disposed right away and their impact on the whole system is minimized. ODMAC has also other components, one of the most interesting being opportunistic forwarding and its close sibling, LAR. Despite effectively being a networking functionality, this can be efficiently implemented inside ODMAC, bringing the two layer closer and starting to blur the dividing line between them. We noticed this being a recur-

ring topic in our design and ideas.

By analyzing ODMAC and RI protocols in general we realized that a very effective new attack was possible, the beacon replay attack. We described this new attack in Chapter 5 and show how it can affect any kind of protocol relying on beaconing techniques. We also showed how preexisting techniques do not apply in this case because of how EH-WSNs behave and function. To address this problem we introduced a new challenge-response protocol called RAP that allows to establish the identity of a receiver. The protocol has two major modes, the detection mode is meant for typical scenarios when the network is not trusted but there are no specific reasons to consider it particularly insecure, and when the application can tolerate it. In this mode the receiver authentication and the data transmission happen simultaneously. As a result a data packet could be given away to an attacker right before realizing his true identity. The advantage of this mode is its low energy cost since it imposes a small overhead in terms of packet size, but no additional message exchange are required. The prevention mode on the other hand is meant to be used when the detection phase has proven the existence of an ongoing attack or when the application requires considerable security. This mode decouples the authentication and the data exchange phases, engaging in the latter only if the former completed successfully. The downside of this is that one extra message has to be exchanged back and forth between the two nodes, thus increasing its cost. We have also discussed about transition policies and when it is useful and appropriate to switch from one mode to the other depending on external parameters imposed by the application.

We then move into the world of adaptive security in Chapter 6. Here we explore the possibility of changing security parameters according to the current amount of energy in a node. Thanks to the EH capabilities the reservoir of a node could be in completely different situations in two moments in time. We present a scheme where each node decides and advertises the security capabilities that it can provide and the requirements of the packets that it is willing to accept. Different security capabilities correspond to different configurations of security parameters such as encryption algorithm, key-length and block-size. We start discussing a static mode

where each node has an immutable preassigned set of capabilities. We use this mode as a tool to introduce the scheme itself and start discussing the dynamic mode. Here nodes with different capabilities coexist for example by being organized into security zones, and the security requirements of a packet can be varied from hop to hop depending on the type of zone that has to be traversed. Finally we introduce the path mode which provides an example of how traffic can be differentiated and sent across different paths according to its importance and its security requirements. We also discuss how well-matched RI protocols are for setting up this kind of infrastructure since the sender can decide a priori whether or not a node can match both the routing and the security constraints of a packet. Finally we argue that, just like we did for RAP, what we call mode is not an hard-coded set of rules, but rather a particular configuration of the nodes that provides the required behavior. This highlights that an high degree of customization is left in the protocol and that it can be tailored to fit and match the constraints dictated by the specific application.

In Chapter 7 we analyzed key establishment and managements schemes. Many aspects of security are often achieved through encryption which is a fundamental tool that relies on secure and sound implementations, and strong keys. We discussed the most common approaches for how to manage multiple keys and why these are needed. We then focused on a specific method called multi-path key reinforcement which takes one key shared between two nodes and aims at reinforcing it by means of composition with additional keys from other nodes and through disjoint paths. We designed and introduced a method that enables nodes of an EH-WSN to select an appropriate number of participants in the protocol, depending on their available energy. We devised a threshold scheme that allows each node to independently define the terms of its participation in the protocol. We also described a fully adaptive scheme where the thresholds are dynamic and move according to the current harvesting rate. This permits high-energy nodes to be available to join more frequently and for longer periods of time, while low-energy nodes can avoid to be exploited. Thanks to the adaptivity, these changes take place without having to set fixed limits that would be only as good as the approximation and

understanding of the system when it is designed, but rather by reacting and adapting to different situations independent of why and how they come to be.

Finally in Chapter 8 we presented an implementation of some of our protocols and schemes. To the best of our knowledge this is one of the first and very few implementations specifically focused on EH-WSNs. We then conducted several experiments that proved our theories and helped us building our confidence that our intuition was correct. This also deepened our understanding of the topic and reminded us how big the difference between theory and practice can be, and how things that are given for granted in the theoretical work are a nightmare to recreate in the real one.

9.1 Future Work

We will now discuss some of the possible directions that can be taken to further expand this work. We feel that much more can be done to improve security in sensor networks, and that taking advantage of EH capabilities can help reaching the goal of achieving more robust and resilient systems. Starting from the security suite developed for ODMAC a further investigation of authenticated-encryption schemes can be taken into account, especially given the latest developments of OCB becoming free to use. A more in-depth analysis could reveal how much energy is necessary to achieve each different property and what can be saved by combining them.

Additional implementation work is a major area to cover. Due to time constraints, protocols such as RAP and the adaptive schemes presented have been conceptualized or evaluated through simulations, where a real implementation would help to present more accurate results. Making use of other platforms for implementation is also a key improvement. Each different type of hardware has a different design and different constraints. Adapting our current design to run on a different kind of nodes to see how the system behaves is a fascinating area to explore. Further experiments can also be done in conjunction with the harvesting hardware, possibly by using solutions that provide a different approach.

Ultimately a global approach can be taken. By designing and defining new different applications, with different requirements, it is possible to study each scheme and see how its parameters can be adapted in order to increase its performance, but also by looking at the system as a whole and making sure that the cooperation level between the different components is optimal and that, where possible, one scheme takes advantage of what another one has to offer.

9.2 Final Remark

To conclude, in this work we had an extensive look at security in EH-WSNs trying to cover different aspects. At the same time we wanted to maintain an organized and structured approach and, where possible, trying to take advantage from what has already been done for regular WSNs and in security in general. We also kept building new protocols and schemes by using our previous work as foundation, making sure that different aspects could work together, and that key goals would be preserved. Probably the most important conclusion at which we arrived is that sensor networks are too specialized and constrained to heave a one-size-fits-all solution. Each application is completely different from another one and so are the requirements. What sounds impossible in one case, is routine in another one. For this reason we strongly believe that two points are essential for developing good sensor network protocols and applications. The first one is moving away from the legacy layer-oriented approach towards an holistic approach. This allows the different components of each node to cooperate in harmony and take advantage not only of the different service they provide, but also of their inner workings. We saw this when we were able to include routing schemes both in ODMAC and in our adaptive security scheme, or when we used beacons to convey information that did not strictly concerned the MAC protocol, but that we could piggyback on them since they had to be sent in any case. The second point is that trying to account for all the possible application requirements and scenarios is hard, error-prone and nigh on impossible. This is why protocols and schemes should not try to predict and solve all the problems, but rather they should provide robust mechanics to ad-

dress core and common issues that are typical of this field. It should be then left to some wise man with good knowledge of the *specific* application to compose those mechanics, optimize the configuration and tweak all the parameters while holding the tongue at the right angle and stroking his long gray beard.

Finally, we believe that this particular field of study is extremely new and that significant progress can still be made. We have the hope and the ambition that this work can somehow help the research community to break new paths and contribute to the overall development of this fascinating topic.

Bibliography

[1] Ian Fuata Akyildiz and Mehmet Can Vuran. *Wireless Sensor Networks*. Ian F. Akylidiz series in Communications and Networking. Wiley, 2010.

[2] AVISPA. Deliverable 2.3: The Intermediate Format, 2003. URL http://www.avispa-project.org/delivs/2.3/d2-3.pdf.

[3] Abdelmalik Bachir, Mischa Dohler, Thomas Watteyne, and Kin Kwan Leung. MAC Essentials for Wireless Sensor Networks. *IEEE Commununications Surveys & Tutorials*, 12(2):222–248, 2010.

[4] David Basin, Sebastian Mödersheim, and Luca Viganò. OFMC: A Symbolic Model Checker for Security Protocols. *International Journal of Information Security*, 4(3):181–208, 2005.

[5] Zinaida Benenson, Andreas Dewald, and Felix Christoph Freiling. Presence, Intervention, Insertion: Unifying Attack and Failure Models in Wireless Sensor Networks. Technical report, University of Mannheim, 2009.

[6] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. *Journal of Cryptology*, 18(4):291–311, 2005.

[7] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 82–96. IEEE, 2001.

[8] Andrey Bogdanov, Lars Ramkilde Knudsen, Gregor Leander, Christof Paar, Andrey Poschmann, Matthew John Barton Robshaw, Yannick Seurin, and Charlotte Vikkelsø. PRESENT: An Ultra-Lightweight Block Cipher. In *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 450–466. Springer, 2007.

[9] Haowen Chan, Adrian Perrig, and Dawn Song. Random Key Predistribution Schemes for Sensor Networks. In *Proceedings of the 22nd IEEE Symposium on Security and Privacy (SP)*, pages 197–213. IEEE, 2003.

[10] Thermo Life Energy Corp., 2014. URL http://www.poweredbythermolife.com.

[11] Cymbet Corporation. Using the EnerChip in Pulse Current Applications (Revision 3), 2008. URL http://www.cymbet.com/pdfs/AN-1025.pdf.

[12] Cymbet Corporation. EnerChip EP Universal Energy Harvester Eval Kit (Revision H), 2014. URL http://www.cymbet.com/pdfs/DS-72-13.pdf.

[13] Cymbet Corporation. EnerChip CC Energy Harvester Evaluation Kit (Revision B), 2014. URL http://www.cymbet.com/pdfs/DS-72-20.pdf.

[14] Jing Deng, Richard Han, and Shivakant Mishra. Limiting DoS Attacks During Multihop Data Delivery in Wireless Sensor Networks. *International Journal of Security and Networks*, 1(3/4):167–178, 2006.

[15] Dorothy Elizabeth Denning and Giovanni Maria Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24(8):533–536, 1981.

[16] Ghislain Despesse, Thomas Jager, Jean-Jacques Chaillout, Jean-Michel Léger, Andrea Vassilev, Skandar Basrour, and Benoît Charlot. Fabrication and Characterization of High Damping Electrostatic Micro Devices for Vibration Energy Scavenging. In *Proceedings of 7th Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP)*, pages 386–390. HAL Inria, 2005.

[17] Alessio Di Mauro, Davide Papini, and Nicola Dragoni. Security Challenges for Energy-Harvesting Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems (PECCS)*, pages 422–425. SciTePress, 2012.

[18] Alessio Di Mauro, Davide Papini, Vigo Roberto, and Nicola Dragoni. Introducing the Cyber-Physical Attacker to Energy-Harvesting Wireless Sensor Networks. *Journal of Networking Technology*, 3:139–148, 2012.

[19] Alessio Di Mauro, Davide Papini, Roberto Vigo, and Nicola Dragoni. Toward a Threat Model for Energy-Harvesting Wireless Sensor Networks. In *Proceedings of the 4th International Conference on Networked Digital Technologies (NDT)*, pages 289–301. Springer, 2012.

[20] Alessio Di Mauro, Xenofon Fafoutis, Sebastian Mödersheim, and Dragoni Nicola. Detecting and Preventing Beacon Replay Attacks in Receiver-Initiated MAC Protocols for Energy Efficient WSNs. In *Proceedings of the 18th Nordic Conference on IT Systems (NordSec)*, pages 1–16. Springer, 2013.

[21] Danny Dolev and Andrew Chi-Chih Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[22] Paul Erdős and Alfréd Rényi. On the Evolution of Random Graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, 5: 17–61, 1960.

[23] Laurent Eschenauer and Virgil Dorin Gligor. A Key-Management Scheme for Distributed Sensor Networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 41–47. ACM, 2002.

[24] Xenofon Fafoutis and Nicola Dragoni. ODMAC: An On-Demand MAC Protocol for Energy Harvesting - Wireless Sensor Networks. In *Proceedings of the 8th ACM Symposium on Performance Evaluation of Wireless Ad-Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, pages 49–56. ACM, 2011.

[25] Xenofon Fafoutis and Nicola Dragoni. Analytical Comparison of MAC Schemes for Energy Harvesting-Wireless Sensor Networks. In *Proceedings of the International Workshop on Algorithms and Concepts for Networked Sensing Systems Powered by Energy Harvesters (EnHaNSS)*, pages 1–6. IEEE, 2012.

[26] Xenofon Fafoutis, Dušan Vučković, Alessio Di Mauro, Nicola Dragoni, and Jan Madsen. Poster Abstract: Energy-Harvesting Wireless Sensor Networks. In *Proceedings of the 9th European Conference on Wireless Sensor Networks (EWSN)*, pages 84–85. University of Trento, 2012.

[27] Xenofon Fafoutis, Alessio Di Mauro, and Nicola Dragoni. Sustainable Medium Access Control: Implementation and Evaluation of ODMAC. In *Proceedings of the 4th Workshop on Energy Efficiency in Wireless Networks and Wireless Networks for Energy Efficiency (E2Nets)*, pages 407–412. IEEE, 2013.

[28] Xenofon Fafoutis, Alessio Di Mauro, and Nicola Dragoni. Sustainable Performance in Energy Harvesting Wireless Sensor Setworks. In *Proceedings of the 4th International Conference on Future Energy Systems (EENERGY)*, pages 267–268. ACM, 2013.

[29] Xenofon Fafoutis, Alessio Di Mauro, Madava Dilshan Vithanage, and Nicola Dragoni. Receiver-Iinitiated Medium Access Control Protocols for Wireless Sensor Networks. *The International Journal of Computer and Telecommunications Networking*, 76:55–74, 2015.

[30] International Organization for Standardization (ISO). *Information Technology – Security Techniques – Lightweight Cryptography – Part 2: Block Ciphers*, 2012.

[31] Nikos Fotiou, Giannis F. Marias, George C. Polyzos, Pawel Szalachowski, Zbigniew Kotulski, Michael Niedermeier, Xiaobing He, and Hermann de Meer. Towards Adaptable Security for Energy Efficiency in Wireless Sensor Networks. In *Proceedings of the 28th meeting of the Wireless World Research Forum (WWRF)*, pages 1–6. Wireless World Research Forum, 2012.

[32] James M. Gilbert and Farooq Balouchi. Comparison of Energy Harvesting Systems for Wireless Sensor Networks. *International Journal of Automation and Computing*, 5(4):334–347, 2008.

[33] Peter Glynne-Jones, Stephen Paul Beeby, and Neil M. White. Towards a Piezoelectric Vibration-Powered Microgenerator. *IEE Proceedings of Science, Measurement and Technology*, 148(2):68–72, 2001.

[34] Peter Glynne-Jones, Michael John Tudor, Stephen Paul Beeby, and Neil M. White. An Electromagnetic, Vibration-Powered Generator for Intelligent Sensor Systems. *Sensors and Actuators A: Physical*, 110(1-3):344–349, 2004.

[35] Alfred Horn. On Sentences Which are True of Direct Unions of Algebras. *Journal of Symbolic Logic*, 16:14–21, 1951.

[36] IEEE. *IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2003.

[37] Texas Instruments Incorporated. MSP430F22x2, MSP430F22x4 Mixed Signal Microcontroller (Revision G), 2012. URL http://www.ti.com/lit/ds/symlink/msp430f2274.pdf.

[38] Texas Instruments Incorporated. MSP430x2xx, Family: User's Guide (Revision J), 2013. URL http://www.ti.com/lit/ug/slau144j/slau144j.pdf.

[39] Texas Instruments Incorporated. CC2500: Low-Cost Low-Power 2.4 GHz RF Transceiver (Revision C), 2014. URL http://www.ti.com/lit/ds/symlink/cc2500.pdf.

[40] Texas Instruments Incorporated. MSP430: Wireless Development Tool, 2014. URL http://www.ti.com/tool/ez430-rf2500.

[41] Vivaksha Jariwala and Devesh Chhabildas Jinwala. Evaluating Galois Counter Mode in Link Layer Security Architecture for Wireless Sensor Networks. *International Journal of Network Security and its Applications*, 2(4): 55–65, 2010.

[42] Chris Karlof and David Wagner. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. *Ad Hoc Networks*, 1(2–3):293–315, 2003.

[43] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 162–175. ACM, 2004.

[44] Joe Kilian and Phillip Rogaway. How to Protect DES Against Exhaustive Key Search. In *Proceedings of the 16th International Cryptology Conference, Advances in Cryptology (CRYPTO)*, pages 252–267. Springer, 1996.

[45] Jongsung Kim and Raphael Chung-Wei Phan. A Cryptanalytic View of the NSA's Skipjack Block Cipher Design. In *Proceedings of the 3rd International Conference and Workshops on Advances in Information and Security Assurance (ISA)*, pages 368–381. Springer, 2009.

[46] Tadayoshi Kohno, Adriana Palacio, and John Black. Building Secure Cryptographic Transforms, or How to Encrypt and MAC, 2003.

[47] Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In *Proceedings of the 18th International Conference on Fast Software Encryption (FSE)*, pages 306–327. Springer, 2011.

[48] Yee Wei Law, Jeroen Doumen, and Pieter Hartel. Survey and Benchmark of Block Ciphers for Wireless Sensor Networks. *ACM Transactions on Senensor Networks*, 2(1):65–93, 2006.

[49] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, and David Culler. TinyOS: An Operating System for Sensor Networks. In *Ambient Intelligence*, pages 115–148. Springer, 2005.

[50] Shiqun Li, Tieyan Li, Xinkai Wang, Jianying Zhou, and Kefei Chen. Efficient Link Layer Security Scheme for Wireless Sensor Networks. *Journal of Information and Computational Science*, 4(2):553–567, 2007.

[51] En-Yi A. Lin, Jan Rabaey, and Adam Wolisz. Power-Efficient Rendez-vous Schemes for Dense Wireless Sensor Networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 3769–3776. IEEE, 2004.

[52] Gavin Lowe. A Hierarchy of Authentication Specifications. In *Proceedings of the 10th Computer Society Foundations Workshop (CSFW)*, pages 31–43. IEEE, 1997.

[53] Perpetuum Ltd., 2014. URL http://www.perpetuum.co.uk.

[54] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Dorin Gligor. MiniSec: A Secure Sensor Network Communication Architecture. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 479–488. ACM, 2007.

[55] David Martins and Hervé Guyennet. Wireless Sensor Network Attacks and Security Mechanisms: A Short Survey. In *Proceedings of the 13th International Conference on Network-Based Information Systems (NBiS)*, pages 313–320. IEEE, 2010.

[56] Marcin Marzencki, Skandar Basrour, Benoît Charlot, Antonio Grasso, Mikaël Colin, and Laurie Valbin. Design and Fabrication of Piezoelectric

Micro Power Generators for Autonomous Microsystems. In *Proceedings of 7th Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP)*, pages 299–302. HAL Inria, 2005.

[57] Ueli M. Maurer and Pierre E. Schmid. A Calculus for Security Bootstrapping in Distributed Systems. *Journal of Computer Security*, 4:55–80, 1996.

[58] Paul D. Mitcheson, Bernard H. Stark, Peng Miao, Eric M. Yeatman, Andrew S. Holmes, and Tim C. Green. *Analysis and Optimisation of MEMS Electrostatic On-Chip Power Supply for Self-Powering of Slow-Moving Sensors*, pages 48–51. University of Minho, 2003.

[59] Sebastian Mödersheim. Algebraic Properties in Alice and Bob Notation. In *Proceedings of 4th International Conference on Availability, Reliability and Security (ARES)*, pages 433–440. IEEE, 2009.

[60] Sebastian Mödersheim. Abstraction by Set-Membership: Verifying Security Protocols and Web Services with Databases. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, pages 351–360. ACM, 2010.

[61] Sebastian Mödersheim and Luca Viganò. Secure Pseudonymous Channels. In *Proceedings of the 14th European Symposium on Research in Computer Security (ESORICS)*, pages 337–354. Springer, 2009.

[62] Roger Michael Needham and Michael D. Schroeder. Authentication Revisited. *ACM SIGOPS Operating Systems Review*, 21(1):7, 1987.

[63] National Institute of Standards and Technology (NIST). *SKIPJACK and KEA Algorithm Specifications*, 1998.

[64] Chulsung Park and Pai H. Chou. AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes. In *Proceeding of the 3rd IEEE International Conference on Sensor Mesh and Ad Hoc Communications and Networs (SECON)*, pages 168–177. IEEE, 2006.

[65] Sylvain Pelissier, Tamma V. Prabhakar, Hirisave Shankaralingaiah Jamadagni, RangaRao Venkatesha Prasad, and Ignas Niemegeers. Providing Security in Energy Harvesting Sensor Networks. In *Proceedings of the IEEE 8th Consumer Communications & Networking Conference (CCNC)*, pages 452–456. IEEE, 2011.

[66] Adrian Perrig, Ran Canetti, Justin Douglas Tygar, and Dawn Song. Efficient Authentication and Signing of Multicast Streams Over Lossy Channels. In *Proceedings of the 21st IEEE Symposium on Security and Privacy (SP)*, pages 56–73. IEEE, 2000.

[67] Adrian Perrig, Robert Szewczyk, Justin Douglas Tygar, Victor Wen, and David Ethan Culler. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(5):521–534, 2002.

[68] Ronald Linn Rivest. The RC5 Encryption Algorithm. In *Proceedings of the 2nd International Workshop on Fast Software Encryption*, Lecture Notes in Computer Science, pages 86–96. Springer, 1994.

[69] Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In *Proocedings of the 10th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 16–31. Springer, 2004.

[70] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A Block-cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Transasctions on Information and System Security*, 6(3):365–403, 2003.

[71] Shad Roundy and Yang Zhang. Toward Self-Tuning Adaptive Vibration-Based Microgenerators. In *Proceedings of SPIE, Smart Structures, Devices, and Systems II*, pages 373–384. SPIE Press, 2005.

[72] Shad Roundy, Paul Kenneth Wright, and Jan Rabaey. A Study of Low Level Vibrations as a Power Source for Wireless Sensor Nodes. *Computer Communications*, 26(11):1131–1144, 2003.

[73] Winston Khoon Guan Seah, Zhi Ang Eu, and Hwee-Pink Tan. Wireless Sensor Networks Powered by Ambient Energy Harvesting (WSN-HEAP) – Survey and Challenges. In *Proceedings of the 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (VITAE)*, pages 1–5. IEEE, 2009.

[74] Christopher Shearwood and Robert B. Yates. Development of an Electromagnetic Micro-Generator. *Electronics Letters*, 33:1883–1884, 1997.

[75] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In

Proceedings of the 13th International Conference on Cryptographic Hardware and Embedded Systems (CHES), pages 342–357. Springer, 2011.

[76] Sujesha Sudevalayam and Purushottam Kulkarni. Energy Harvesting Sensor Nodes: Survey and Implications. *IEEE Communications Surveys & Tutorials*, 13(3):443–461, 2011.

[77] Yanjun Sun, Omer Gurewitz, and David B. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty CycleMAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proceedings of the 6th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–14. ACM, 2008.

[78] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE: A Lightweight Block Cipher for Multiple Platforms. In *Proceedings of the 19th International Conference on Selected Areas in Cryptography (SAC)*, pages 339–354. Springer, 2013.

[79] Antonio Vincenzo Taddeo, Marcello Mura, and Alberto Ferrante. QoS and Security in Energy-Harvesting Wireless Sensor Networks. In *Proceedings of the 7th International Conference on Security and Cryptography (SECRYPT)*, pages 1–10. IEEE, 2010.

[80] Chryssanthi Taramonli, Roger Julian Green, and Mark Stephen Leeson. Energy Conscious Adaptive Security Scheme for Optical Wireless. In *Proceedings of the 14th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4. IEEE, 2012.

[81] Roberto Vigo. The Cyber-Physical Attacker. In *Computer Safety, Reliability, and Security*, pages 347–356. Springer, 2012.

[82] Christopher M. Vigorito, Deepak Ganesan, and Andrew G. Barto. Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks. In *Proceedings 4th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 21–30. IEEE, 2007.

[83] Arvinderpal Singh Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks. In *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 324–328. IEEE, 2005.

[84] Yong Wang, G. Attebury, and B. Ramamurthy. A Survey of Security Issues in Wireless Sensor Networks. *IEEE Communications & Surveys Tutorials*, 8 (2):2–23, 2006.

[85] Christoph Weidenbach, Renate A. Schmidt, Thomas Hillenbrand, Rostislav Rusev, and Dalibor Topic. System Description: Spass Version 3.0. In *Proceedings of the 21st International Conference on Automated Deduction (CADE)*, pages 514–520. Springer, 2007.

[86] Xueying Zhang, Howard M. Heys, and Cheng Li. Energy Cost of Cryptographic Session Key Establishment in a Wireless Sensor Network. In *Proceedings of the 6th International Conference on Communications and Networking in China (CHINACOM)*, pages 335–339. IEEE, 2011.

[87] Jun Zheng and Abbas Jamalipour. *Wireless Sensor Networks, A Networking Perspective*. Wiley, 2009.