Architectures and Protocols for Secure and Energy-Efficient Integration of Wireless Sensor Networks with the Internet of Things

Malisa Vucinic

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Mališa VUČINIĆ

Thèse dirigée par **Bernard Tourancheau**
et codirigée par **Franck Rousseau et Laurent Damon**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG) et de STMicroelectronics France**
et de l'**École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Architectures and Protocols for Secure and Energy-Efficient Integration of Wireless Sensor Networks with the Internet of Things

Thèse soutenue publiquement le **17 novembre 2015**,
devant le jury composé de :

Mme Nathalie MITTON
Chargée de recherche, Inria Lille–Nord Europe, Rapporteur
M Yacine CHALLAL
Maître de Conférences, Université de Technologie de Compiègne, Rapporteur
M Enzo MINGOZZI
Professeur, University of Pisa, Examinateur
M Fabrice VALOIS
Professeur, INSA Lyon, Président
M Thiemo VOIGT
Professeur, Swedish Institute of Computer Science, Examinateur
M Bernard TOURANCHEAU
Professeur, Université Joseph Fourier, Directeur de thèse
M Franck ROUSSEAU
Maître de Conférences, Grenoble INP, Co-Encadrant de thèse
M Laurent DAMON
Ingénieur, STMicroelectronics, Co-Encadrant de thèse

Abstract

Our research explores the intersection of academic, industrial and standardization spheres to enable secure and energy-efficient Internet of Things. We study the performance of cryptographic primitives on commodity hardware and observe that hardware accelerators reduce execution times by as much as two orders of magnitude. Cryptographic overhead is, however, only one of the factors that influence the overall performance in the networking context. To understand the energy – security tradeoffs, we evaluate the effect of link-layer security features on the performance of Wireless Sensor Networks. We show that for practical applications and implementations, link-layer security features introduce a degradation on the order of a couple of percent, that is often acceptable even for the most energy-constrained systems, such as those based on harvesting.

Because link-layer security puts trust on each node on the communication path consisted of multiple, potentially compromised devices, we protect the information flows by end-to-end security mechanisms. We therefore consider Datagram Transport Layer Security (DTLS) protocol, the Internet standard for end-to-end security in the Internet of Things and contribute to the debate in both the standardization and research communities on the applicability of DTLS to constrained environments. We provide a thorough performance evaluation of DTLS in different duty-cycled networks through real-world experimentation, emulation and analysis. Our results demonstrate surprisingly poor performance of DTLS in networks where energy efficiency is paramount. Because a DTLS client and a server exchange many signaling packets, the DTLS handshake takes between a handful of seconds and several tens of seconds, with similar results for different duty cycling protocols.

But apart from its performance issues, DTLS was designed for point-to-point communication dominant in the traditional Internet. The novel Internet of Things standard, Constrained Application Protocol (CoAP) was tailored for constrained devices by facilitating asynchronous application traffic, group communication and absolute need for caching. The security architecture based on DTLS is, however, not able to keep up and advanced features of CoAP simply become futile when used in conjunction with DTLS. We propose an architecture that leverages the security concepts both from content-centric and traditional connection-oriented approaches. We rely on secure channels established by means of DTLS for key exchange, but we get rid of the notion of "state" among communicating entities by leveraging the concept of object security. We provide a mechanism to protect from replay attacks by coupling the capability-based access control with network communication and CoAP header. Our Object Security Architecture for the Internet of Things (OSCAR) intrinsically supports caching and group communication, and does not affect the radio duty cycling operation of constrained devices. Concepts from OSCAR have already found their way towards the Internet standards and are widely discussed as potential solutions for standardization.

Keywords: Internet of Things, Wireless Sensor Networks, network performance, object security, security architecture, energy efficiency.

Résumé

Nos recherches se situent à l'intersection des sphères académiques et industrielles et des organismes de standardisation pour permettre la mise en place d'un Internet des objets (IoT) sécurisé et efficace. En premier lieu, nous constatons que l'accélération matérielle des algorithmes de cryptographie est nécessaire pour les équipements formant l'IoT car elle permet une réduction de deux ordres de grandeur des durées de calcul. Le surcoût des opérations cryptographiques n'est cependant qu'un des facteurs qui gouverne la performance globale dans le contexte des systèmes en réseau. Nous montrons à travers l'implementation d'applications pratiques que les dispositifs de sécurité de la couche liaison de données n'augmentent que de quelques pour cents la dépense énergétique totale. Ceci est acceptable, même pour les systèmes les plus contraints, comme ceux utilisant la recuperation d'énergie ambiante.

La sécurité de la couche liaison de données contraint de faire confiance à chacun des noeuds du chemin de communication comprenant potentiellement des éléments malveillants. Nous devons donc protéger le flux de données par un mécanisme de bout en bout. Nous étudions le protocole DTLS, standard pour la sécurité de l'IoT. Nous contribuons aux discussions sur l'intérêt de DTLS dans les environnements contraints, à la fois dans les organismes de standardisation et de recherche. Nous évaluons DTLS de manière étendue avec différents réseaux à rapport cyclique ou *duty cycle*, au travers d'expérimentations, d'émulations et d'analyses. De manière surprenante, nos résultats démontrent le coût prohibitif de DTLS dans ces réseaux où l'efficacité énergétique est primordiale. Comme un client et un serveur DTLS échangent beaucoup de paquets de signalisation, la connection DTLS prends entre quelques secondes et quelques dizaines de secondes, dans chacun des protocoles à rapport cyclique étudié.

DTLS a été conçu pour les communications de bout en bout dans l'Internet classique, contrairement au nouveau standard de l'IoT, le protocol CoAP qui est destiné à des machines contraintes, facilite le traffic asynchrone et les communications de groupe et autorise le stockage intermédiaire. Donc, en plus du problème de performance, l'architecture de sécurité basée sur DTLS n'est pas capable de répondre aux contraintes de l'IoT et CoAP devient inutilisable. Nous proposons une architecture qui s'appuie à la fois sur une approche centrée sur le contenu et sur la notion classique de connection. L'échange des clefs est fait à travers des canaux sécurisés établis par DTLS, mais la notion d'états entre les entités de communication est supprimée grâce au concept d'objets sécurisés. Le mécanisme proposé résiste aux attaques par rejeu en regroupant les capacités de contrôle d'accès avec les en-têtes de communication CoAP. Notre architecture à objets sécurisés (OSCAR), supporte intrinsèquement les communications de groupe et le stockage intermédiaire, sans perturber le fonctionnement à rapport cyclique de la radio des équipements contraints. Les idées d'OSCAR sont considerées par les groupes de standardisation de l'Internet en vue d'être intégrées dans les standards à venir.

Mots-clés : Internet des objets, réseaux de capteurs sans fil, performance du réseau, objets sécurisés, architecture de sécurité, efficacité énergétique.

Acknowledgments

The chance to work in the context of an academic and industrial partnership was of an immense importance to a young researcher. Under a CIFRE grant, I was able to pursue my thesis work in collaboration with *STMicroelectronics* and *Laboratoire d'Informatique de Grenoble*. For that, I gratefully acknowledge the funding provided by the *Association Nationale de la Recherche et de la Technologie (ANRT)* who partially supported the collaboration.

The manuscript was valuably improved thanks to the reviewers *Dr. Nathalie Mitton* and *Dr. Yacine Challal*. I thank them both for dedicating their time to review the dissertation and providing helpful comments and constructive feedback. I also thank the examiners *Prof. Enzo Mingozzi* and *Prof. Thiemo Voigt*, as well as the president of the jury *Prof. Fabrice Valois* for all the insightful questions and lengthy discussions during the oral defense.

My deepest gratitude goes to *Bernard*, who has unmistakably led me since the first day of my Master's thesis and throughout this amazing PhD endeavor. I appreciate his dedication of time, contribution of ideas and the moral support that have kept me going even through the most difficult of times. He always ensured I had enough of his support and personal freedom to discover the paths that span much further than the end of my PhD. For this, I am forever grateful.

Over the course of the thesis, I was lucky to have co-advisors in both the academia and the industry. I thank *Laurent* for his support and patience. I thank *Franck* for inspiring me to focus on object security and for being there to brainstorm exciting research tracks. Although not officially a co-advisor, *Roberto* went above-and-beyond to guide me through the industrial sphere. I thank him for influencing the practical approach of my research.

I am grateful to all the faculty, students and post-docs in the Drakkar team for valuable discussions, and particularly to *Prof. Andrzej Duda* for having established the collaboration with STMicroelectronics.

The work within the GreenNet team of STMicroelectronics has made me a better engineer, and I thank all the members and fellow CIFRE students for discussions and the great time we had while preparing for various demonstrations.

During my thesis, I had the privilege to carry out a part of the research at two educational institutions. The stay at the University of New South Wales was made possible thanks to *Prof. Sanjay Jha*. I am thankful to *Pascal Urard* for having initiated my research visit of UC Berkeley, and to *Dr. Thomas Watteyne* and *Prof. Kris Pister* for making it happen. I thank *Prof. Pister* for his influence on how the details are as important as the big picture and *Thomas* for helping me find future challenges.

I am thankful to *Dr. Ludwig Seitz* and *Dr. Göran Selander* on the opportunity to participate in the ACE standardization activities. I also thank *Ludwig* for his thorough feedback on OSCAR that improved the extended version of the paper.

I am grateful for help to all the undergraduate students I had the privilege to supervise, and most notably to *Robin*, *Laurène*, *Titouan* and *Baptiste*.

Thank you, my friends, for making the PhD life rememberable, for every pint at O'Callaghan's, every sample of yet another French wine and each variant of the infamous *tiramisù*.

My parents and my sister have always been a source of inspiration and motivation throughout the years. Thank you for giving me the foundation to meet life and for being the beacon light of my life journey.

Fortunate to have been accompanied by *Bojana*, I admire her patience and envy her endurance for the late hours at work and the time abroad. You have helped me get over all the obstacles along the way and made every step less burdensome. Thank you.

Contents

List of Figures

List of Tables

List of Acronyms

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks

6TiSCH IPv6 over the TSCH mode of IEEE 802.15.4e

AAA Authentication, Authorization, and Accounting

ACE Authentication and Authorization for Constrained Environments

ACK Acknowledgment

ADC Analog to Digital Converter

AEAD Authenticated Encryption with Associated Data

AES Advanced Encryption Standard

AP Access Point

API Application Programming Interface

AS Authorization Server

ASH Auxiliary Security Header

ASN.1 Abstract Syntax Notation One

ASN Absolute Slot Number

BI Beacon Interval

BO Beacon Order

BSL Bootstrap Loader

BTLE Bluetooth Low Energy

C Client

CAP Contention Access Period

CBC-MAC Cipher Block Chaining Message Authentication Code

CBC Cipher Block Chaining

CBOR Concise Binary Object Representation

CCM Counter Mode Encryption and Cipher Block Chaining Message Authentication Code

CCM* Extension of Counter Mode Encryption and Cipher Block Chaining Message Authentication Code

CI Check Interval

CMS Cryptographic Message Syntax

CoAP Constrained Application Protocol

CORE Constrained RESTful Environments

COSE CBOR Object Signing and Encryption

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

CTR Counter

DAC Digital to Analog Converter

DAG Directed Acyclic Graph

DAO DODAG Destination Advertisement Object

DCAF Delegated CoAP Authentication and Authorization Framework

DES Data Encryption Standard

DH Diffie-Hellman

DICE DTLS In Constrained Environments

DIO DODAG Information Object

DIS DODAG Information Solicitation

DLP Discrete Logarithm Problem

DNS Domain Name System

DNSSEC Domain Name System Security Extensions

DODAG Destination Oriented Directed Acyclic Graph

DoS Denial of Service

DSA Digital Signature Algorithm

DSME Deterministic and Synchronous Multichannel Extension

DSSS Direct Sequence Spread Spectrum

DTLS Datagram Transport Layer Security

ECB Electronic Codebook

ECC Elliptic Curve Cryptography

ECDH Elliptic Curve Diffie-Hellman

ECDLP Elliptic Curve Discrete Logarithm Problem

ECDSA Elliptic Curve Digital Signature Algorithm

EUI-64 64-bit Extended Unique Identifier

FFD Full Function Device

GCM Galois/Counter Mode

GPS Global Positioning System

HTTP Hypertext Transfer Protocol

HVAC Heating, Ventilation, and Air Conditioning

IC Integrated Circuit

ICMPv6 Internet Control Message Protocol version 6

IE Information Elements

IETF Internet Engineering Task Force

IoT Internet of Things

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISM Industrial, Scientific, Medical

IV Initialization Vector

JCE Join Coordination Entitiy

JOSE JSON Object Signing and Encryption

JSON Javascript Object Notation

JTAG IEEE 1149.1 JTAG

LED Light Emitting Diode

LNA Low Noise Amplifier

LPM Low Power Mode

LRP Lightweight Routing Protocol

LRU Least Recently Used

LWM2M Lightweight Machine to Machine

M2M Machine to Machine

MAC Medium Access Control

MCU Microcontroller Unit

MEMS Microelectromechanical Sensors

MGCP Media Gateway Control Protocol

MIC Message Integrity Code

MITM Man in the Middle

MOP Mode of Operation

MTU Maximum Transmission Unit

NFC Near Field Communication

NSA National Security Agency

O-QPSK Offset Quadrature Phase-Shift Keying

OS Operating System

OSCAR Object Security Architecture for the Internet of Things

OSCOAP Object Security for CoAP

OTP One Time Pad

PA Power Amplifier

PAN Personal Area Network

PCB Printed Circuit Board

PDR Packet Delivery Ratio

PDU Protocol Data Unit

PGP Pretty Good Privacy

PMU Power Management Unit

PSK Pre-Shared Key

PV Photovoltaic

RC4 Rivest Cipher 4

RDC Radio Duty-Cycling

REMBASS Remote Battlefield Sensor System

REST Representational State Transfer

RFD Reduced Function Device

ROLL Routing Over Low-Power and Lossy Networks

RPL IPv6 Routing Protocol for Low-Power and Lossy Networks

RS Resource Server

RSA Rivest, Shamir, Adelman

RTP Real Time Protocol

S/MIME Secure/Multipurpose Internet Mail Extensions

SBP Systematic Beacon Payload

SD Superframe Duration

SEP Smart Energy Profile

SFD Start of Frame Delimiter

SIP Session Initiation Protocol

SO Superframe Order

SoC System on Chip

SPI Serial Peripheral Interface

SSH Secure Shell

SSL Secure Sockets Layer

ST STMicroelectronics

TCP Transmission Control Protocol

TLS Transport Layer Security

TLV Type-Length-Value

TSCH Time-Slotted Channel Hopping

UDG Unit Disk Graph

UDP User Datagram Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

USB Universal Serial Bus

VPN Virtual Private Network

WSN Wireless Sensor Network

XOR Exclusive OR

Contributions

Journal Publications

[c_1] **Mališa Vučinić**, Bernard Tourancheau, Franck Rousseau, Andrzej Duda, Laurent Damon, and Roberto Guizzetti. OSCAR: Object security architecture for the Internet of Things. *Elsevier Ad Hoc Networks*, 32:3–16, 2015.

[c_2] Liviu-Octavian Varga, Gabriele Romaniello, **Mališa Vučinić**, Michel Favre, Andrei Banciu, Roberto Guizzetti, Christophe Planat, Pascal Urard, Martin Heusse, Franck Rousseau, Olivier Alphand, Etienne Duble, and Andrzej Duda. GreenNet: an Energy Harvesting IP-Enabled Wireless Sensor Network. *IEEE Internet of Things Journal*, 2015.

[c_3] Savio Sciancalepore, **Mališa Vučinić**, Giuseppe Piro, Gennaro Boggia, and Thomas Watteyne. Link-layer Security in TSCH Networks: Effect on Slot Duration. *Submitted for Publication*, 2015.

International Conferences

[c_4] **Mališa Vučinić**, Bernard Tourancheau, Franck Rousseau, Andrzej Duda, Laurent Damon, and Roberto Guizzetti. OSCAR: Object security architecture for the Internet of Things. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, pages 1–10, June 2014.

[c_5] **Mališa Vučinić**, Bernard Tourancheau, Thomas Watteyne, Franck Rousseau, Andrzej Duda, Roberto Guizzetti, and Laurent Damon. DTLS Performance in Duty-Cycled Networks. In *Personal, Indoor, and Mobile Radio Communication (PIMRC), 2015 IEEE 26th Annual International Symposium on*, pages 1–6, August 2015.

[c_6] **Mališa Vučinić**, Bernard Tourancheau, Franck Rousseau, Andrzej Duda, Laurent Damon, and Roberto Guizzetti. Energy cost of security in an energy-harvested IEEE 802.15.4 Wireless Sensor Network. In *Embedded Computing (MECO), 2014 3rd Mediterranean Conference on*, pages 198–201, June 2014.

[c_7] Titouan Coladon, **Mališa Vučinić**, and Bernard Tourancheau. Multiple Redundancy Constants with Trickle. In *Personal, Indoor, and Mobile Radio Communication (PIMRC), 2015 IEEE 26th Annual International Symposium on*, pages 1–6, August 2015.

[c_8] **Mališa Vučinić**, Gabriele Romaniello, Laurene Guelorget, Bernard Tourancheau, Franck Rousseau, Olivier Alphand, Andrzej Duda, and Laurent Damon. Topology construction in RPL networks over beacon-enabled 802.15.4. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–7, June 2014.

Standardization

[c_9] Ludwig Seitz, Göran Selander, and **Mališa Vučinić**. Authorization for Constrained RESTful Environments. draft-seitz-ace-core-authz-00. Work in progress, June 2015.

National Conferences in French

[c_{10}] Titouan Coladon, **Mališa Vučinić**, and Bernard Tourancheau. Improving trickle fairness: Locally-calculated redundancy constants. In *Nouvelles Technologies de la Répartition (NOTERE), 2015*, pages 1–6, July 2015.

Software Development Projects

[c_{11}] **Mališa Vučinić**, Robin Stieglitz. IEEE 802.15.4 security for Contiki and GreenNet boards. *Internal repository of STMicroelectronics* – GreenNet, April–August 2013.

[c_{12}] **Mališa Vučinić**. Generic object security library for Contiki with support for encrypted, signed and encrypted-signed objects. Binding of the library with CoAP to implement OSCAR for constrained devices. Porting of tinyDTLS to GreenNet Contiki and beacon-enabled IEEE 802.15.4. *Internal repository of STMicroelectronics* – GreenNet, June 2013 – March 2014.

[c_{13}] **Mališa Vučinić**. CRYPTO ENGINE API for hardware-accelerated symmetric cryptographic primitives and implementations for TelosB, CC2538 and GreenNet boards. *Official OpenWSN repository*, February–April 2015.

[c_{14}] Savio Sciancalepore, Giuseppe Piro, Gennaro Boggia, Luigi Alfredo Grieco, **Mališa Vučinić**. IEEE 802.15.4e TSCH security for OpenWSN. *Official OpenWSN repository*, May–June 2015.

[c_{15}] **Mališa Vučinić**, Elodie Morin, Liviu-Octavian Varga, Iacob Juc. Port of GreenNet boards to OpenWSN stack (IEEE 802.15.4e TSCH). *Internal repository of STMicroelectronics* – GreenNet, June–July 2015.

Chapter 1

Introduction

Our civilization has been irreversibly affected by the advent of the Internet. Enabled by the global connectivity, the unprecedented means for knowledge sharing has changed how we interact with the surroundings. Computing platforms have permitted to us, humans, to engage in this global network of information and benefit from knowledge, accumulated over millenniums of human history.

But we are witnessing a new era. An era where the information is no longer required to be produced by humans. The technology is providing means to extend civilization's nervous system, the Internet, to our environment. We have become able to coordinate our actions on local, national or global level, according to the real-time inputs from the physical surroundings.

For the first time, we are able to make decisions grounded in the environmental feedback. We expect our technology to play a crucial role towards the sustainable planet. Today, because there are already 15 billion devices connected in 2015 [182] and we expect an exponential growth in the future.

Billions of devices that sense and actuate on the environment form the *Internet of Things (IoT)*. These devices serve very different purposes, ranging from daily rechargeable smart watches, mains-powered electricity meters in our house, to wireless sensors and actuators monitoring the production of a power plant. Some can talk directly to our smartphones while others form Wireless Sensor Networks (WSNs), an infrastructure of interconnected devices that can *sense* and *actuate* on the environment. In this manuscript, we consider the Internet of Things formed of Internet-integrated Wireless Sensor Networks, whether the integration applies to a device, or the data it produces.

1.1 Enabling Technologies

Like many others, technologies behind Wireless Sensor Networks were first envisioned for military applications. All the way back in 1967 [115], Remote Battlefield Sensor System (REMBASS) was envisioned to detect battlefield activities in real-time and transmit the information towards the control center, through various radio repeaters.

The omnipresent wave of interest in WSN and IoT technology was triggered by the Smart Dust project [122] proposed in 1997 [115] and led by Prof. Kris Pister at the University of California, Berkeley, where the author had the privilege of carrying out part of the research leading to this manuscript. Smart Dust aimed at designing

a cubic millimeter device with a sensor, power supply, analog circuitry, bidirectional optical communication and a programmable microprocessor [120].

The Smart Dust project attracted the interest of Computer Science community and many proposals followed, albeit leveraging *macro* motes and commercial off-the-shelf components. Nevertheless, the commercial interest was recognized soon and various companies emerged. These companies had in one form or another proprietary hardware and communication stacks, and it was quickly recognized that having a multitude of proprietary systems connected to the Internet impedes the much hoped-for scalability and explosion of the Internet of Things [115].

Standards followed. In 2003, IEEE 802.15.4 standard was published and it specified the requirements of a low-power radio transceiver and a corresponding medium access protocol. IEEE 802.15.4 served as the base of early wireless sensing and automation solutions, like ZigBee 1.0 and ZigBee 2006 [185]. Even today, the IEEE 802.15.4 standard is the main solution for the interconnection of low-power devices with sensing capabilities into a Wireless Sensor Network. The integration of such Wireless Sensor Networks with the Internet has been handled by Internet Engineering Task Force (IETF), since 2007 when first IoT standards emerged.

1.2 The Current Status

Today, it is a daunting task to precisely identify the IoT use cases. Smart wearables are often attributed as IoT products but due to their daily rechargeable batteries, they pose technical challenges similar to smartphones, which we do not consider in this thesis. Arguably the first commercial application of WSN technologies was the industrial automation, where real-time sensor readings and remote actuation facilitate the optimization of the production process. There are reports of at least 9200 such proprietary WSNs, clocking over 987 million operating hours and deployed on all continents [115]. Remote metering is also a good example, and electricity, water or gas providers are already providing services to users to monitor their consumption. This happens at granularity levels much finer than once a month or even a year, as it had been common once, thanks to smart meters installed at users' premises. Home automation solutions have also been around for years, but the factor of interoperability is much more a determining factor of success with a common user than with a commercial enterprise in charge of an industrial plant or a network of water meters. Certainly, householders rarely want to commit to the vendor of "smart" lightbulbs to also be in control of every other smart appliance. Smart city solutions controlling street lightning, monitoring traffic, pollution levels or available parkings are also increasingly deployed in cities like Grenoble, Barcelona, Santander, Moscow, Tokyo, ...

But we are waiting for their convergence, enabled by *interoperability*. Convergence that will permit smart electric grids to operate on inputs from users' households, smart cities to operate on inputs from circulating vehicles, and users, us, to leverage the available information and improve our daily lives. The improvement

to our lives can be regarded as a more efficient commute or a fresher air that we breath.

1.3 The Dark Side

As does any technology, Internet of Things opens up new opportunities that can be leveraged for the greater good, or the individual benefit. Oftentimes, the latter leads to malicious actions that serve as the base motivation for the entire technical discipline of computer security.

In the Internet of Things context, every sensor reading, no matter how benign it may appear, is an input to a decision process. A temperature reading represents much more than a pure physical value. It is a contextual information that may be acted upon in various scenarios, from nuclear power plants to smart coffee machines. Unauthorized modification of a couple of bytes of data *is* equivalent to someone breaking in your house or the nearby nuclear power plant, through the unlocked coffee machine.

Then, there is always the Orwellian path of surveillance. The path we seem to be headed to, for quite some time now, with or without the Internet of Things. But we believe that the Internet of Things can actually be a catalyst for change. We might finally understand that our data is, well, ours. And because Internet of Things is all about the data, we may start using solutions that allow us to protect the *data*. Not only to "securely" give the data away.

Our research explores the protection mechanisms and their technical costs for keeping the dark side consequences at minimal. Where existing solutions prioritize the protection of *means*, that is the communication, rather than the protection of *content*, we pursue the path of data-centric paradigms to provide finer control over the user data.

1.4 Technical Challenges

The scale of the Internet of Things poses engineering challenges at many levels. The basic requirement is that of low-cost, because mass market adoption is dependent on product affordability. Since we consider hundreds of devices per user, the unit-cost must be kept low which is reflected in the available hardware components. Practically, this means much less computational power than what we are used to on personal computers and smartphones, with system clocks on the order of a megahertz, and very limited memory storage, tens of kilobytes for volatile, and hundreds of kilobytes for non-volatile memory.

But the economic cost is not the only factor driving device design. Our devices are typically powered by a single battery, throughout their lifetime. Since the market requires the lifetime to be on the order of several years, we use both hardware and software techniques to minimize average current draws. Available hardware provides fundamental limits while software techniques allow us to maximize the

lifetime while providing the desired functionality. Achieving energy-efficiency is not only important from the point of view of a single device. Hundred billions of devices pose global challenges on the availability of raw energy needed to power them or raw materials needed to build enough batteries. Because we do not want the cost of rolling out the Internet of Things technology to overweight its benefits for the environment, we strongly believe that devices based on energy harvesting should be the *future*. Security and networking mechanisms that we explore in this manuscript are inspired by such a device – the GreenNet mote – and aim at enabling a sustainable, interoperable and secure Internet of Things.

1.5 Manuscript and Contribution Overview

Contributions of this manuscript lie on the intersection of academic, standardization and industrial spheres of security and networking. We evaluate standards-based solutions in real-world, energy-constrained scenarios and draw conclusions on their applicability and potential issues. Once we identify critical aspects, either from performance or capability point of view, we propose novel mechanisms and architectures, rooted in the problematics of achieving 1) energy-efficiency for prolonging device lifetime and local sustainability; 2) interoperability to enable convergence of different technologies.

In Part I, we provide the background on the internals of a single device (Chapter 2) and communication standards that we consider in this thesis (Chapter 3). In Chapter 4, we continue by overviewing the basic building blocks of security solutions – the cryptographic primitives – and focus on algorithms typically used in the context of the Internet of Things. We illustrate the performance of these primitives in the context of a single device by discussing the first, not yet published, contribution of this manuscript: An application programming interface that leverages the hardware-software implementation of symmetric cryptographic primitives, together with performance benchmarks for three different types of IoT devices.

Part II considers the security of Internet-integrated Wireless Sensor Networks. We discuss the typical threats and state the art in securing the considered communication stack in Chapter 5. We first evaluate security standards in charge of protecting direct radio communication between two devices [c_3] [c_6] in Chapter 6. We then proceed to the evaluation [c_5] of the standard end-to-end security mechanism in Chapter 7, where we identify both performance and capability issues. We tackle these problems in Chapter 8 by proposing a system-level architecture [c_4] [c_1] for protecting the IoT *data*.

Part III and Chapter 9 summarize the results and discuss future perspectives.

In order not to distract the reader from security-related material of this manuscript, we present in the Appendix two contributions [c_8] [c_7] related to energy-efficient construction and maintenance of the network. The content of the Appendix is independent of Part II and can be followed after Part I.

Part I

Internet of Constrained Devices and Cryptography

Chapter 2

Constrained Hardware

This chapter describes the internals of an IoT device. We provide some basic terminology in Section 2.1. In Section 2.2, we give an overview of typical IoT hardware by discussing the blocks that make an IoT device *smart*. We introduce the GreenNet project of STMicroelectronics (ST) and the internal structure of GreenNet boards in Section 2.3. We conclude the chapter in Section 2.4.

2.1 Terminology

We interchangeably use terms *constrained device*, *device*, *smart object*, *thing*, *board*, and *platform* to denote a Printed Circuit Board (PCB) that contains different electronic components, e.g. Integrated Circuit (IC), energy supply, and is capable to participate in the IoT. Once such a device becomes part of a network, it is referred to as a *mote* or simply a *node*.

2.2 Building Blocks of a *Thing*

Analyzing, sensing and communicating are the three "cognitive" functions that make a device *smart*. They are reflected in hardware that we are witnessing today.

Microcontroller Unit

Microcontroller Unit (MCU) is the brain of a device. In the most fundamental setting, it integrates a processor, program memory (e.g. flash) and data memory (e.g. RAM).

Digital logic within the MCU is driven by clock ticks generated by a crystal oscillator that beats at a precise frequency. As MCU may draw a substantial amount of current when in normal processing (i.e. active mode), typical design allows several modes that a user can leverage. These *sleep* modes differ in whether data memory is retained or not, or whether internal peripherals are still active while the rest of the MCU is asleep.

Requirements on low-power drive memory sizes and processor capabilities. Larger memory sizes imply larger cell and transistor count which directly influences leakage currents. Consequently, MCU sleep modes that retain data memory have higher power consumption in respect to those where data memory is not retained. We list some typical examples and their characteristics in Table 2.1.

Table 2.1: Examples of commercially available MCUs and their characteristics.

Part Number	Instruction Size [bits]	RAM [kB]	CPU on [mA/ MHz]	CPU sleep [μA]
STM32L151 [148]	32	32	0.185	0.44
LTC5800-IPM SoC [94]	32	72	0.176	0.8
CC2538 SoC [157]	32	32	0.438	0.4
MSP430F1x [158]	16	10	1.8	5.1
ATmega128L [8]	8	4	1.25	<15

Once MCU is asleep, it can be woken up by an interrupt. The interrupt may be generated either by 1) an external chip or 2) by MCU itself. As an example of (1), MCU may be woken up by an interrupt from an accelerometer detecting unusual acceleration pattern and thus signaling an alarm, or a pass to zero and thus signaling a free fall. Case (2) may be due to the expiry of a hardware *timer* internal to the MCU. Timer is a simple counter that increments with each clock tick. When this counter reaches a value predefined in the corresponding comparator register, an interrupt is raised. By configuring the comparator register, the programmer can decide how long the MCU should sleep.

In MCU sleep modes, memory leakage currents and the active crystal oscillator are the main contributors to the overall consumption. Low consumption in sleep mode is of utmost importance as IoT devices spend more than 99.99% of time sleeping.

Sensors and Actuators

As humans, we use our senses to interact with the environment. IoT devices reach out into the physical world by integrating hardware sensors and actuators that can either be queried by MCU for a physical reading or instructed to change a physical quantity. Advances in Microelectromechanical Sensors (MEMS) manufacturing technology have reduced their cost and size and effectively enabled IoT [174]. Examples of commonly found sensors include: temperature, humidity, light, pressure, carbon monoxide, carbon dioxide, accelerometer, gyroscope and on higher-end devices a Global Positioning System (GPS) receiver.

Sensor design leverages different physical phenomena and material properties in order to provide an easily measurable output such as voltage or current. For instance, one can design a temperature sensor by using a temperature-sensitive resistor and measuring the output voltage over constant input current. Light sensor can be a simple p-n junction, i.e. a diode, whose bandgap energy corresponds to the visible light spectrum and the absorbed photons cause current flow. As this

current is related to the incident light, it is possible to estimate the light intensity the sensor is exposed to. Depending on their proper hardware design, sensor circuits can provide an analog or a digital output. In the analog case, output pin of the sensor is connected to an Analog to Digital Converter (ADC) that MCU can read and provide the programmer with a digital value. In the digital case, the sensor circuitry directly provides a digital value as sensor output.

IoT devices can also be used to control the environment. By managing the output voltage of a relay[1], one can for example dim or brighten a "smart" light bulb, lock or unlock a door, control the aperture of different valves.

Radio Transceiver

Human civilization as a whole owes its progress to a seemingly simple evolutive capability – transfer of knowledge from one entity to the other, current generation to the next. Sensing and local processing of a physical quantity do not allow us to exploit the information in a wider context. In the IoT world, knowledge is transferred and diffused through a communication interface. Because wired interfaces are simply too expensive for billions of devices and often not practical[2] we use a radio transceiver.

The basic role of a radio transceiver is on one hand to convert digital signal into an electromagnetic wave that can propagate in free space. On the other hand, it needs to be able to interpret the received signal into a meaningful information. When it comes to electromagnetic propagation, fundamental physics drives power loss between a transmitter and a receiver.

Before transmitter can emit the signal in the air, digital stream of data called radio *frame*, needs to be encoded and converted to an analog signal using a modulation scheme. Power Amplifier (PA) amplifies the analog signal before it is radiated by an antenna and extends the range where the signal can be received. Output power of PA can be configured by the programmer and for low power chips it typically ranges from -50dBm to 10dBm, with most commonly used setting of 0dBm (1mW).

On the receiver side, if the signal picked up by the antenna is weaker than the theoretical minimum, it is impossible to differentiate signal from noise and obtain a meaningful information. For example, with 2MHz wide channels and coding gains of IEEE 802.15.4 standard, theoretical minimum is approximately -113.2dBm [87]. That means that the *theoretically* weakest received signal must be greater than -113.2dBm. Design techniques of radio transceivers influence the weakest signal that can be successfully received by a certain radio. That measure is called *sensitivity* and for commercially available radios with 2MHz wide channels is around -90dBm [115]. Receiver has no means of detecting a transmission other than to continuously amplify the input signal from the antenna using Low Noise Ampli-

[1] Relay is a simple electronic circuit that switches high voltage or current using a low power command circuit, such as the output pin of a MCU.

[2] For example in rotating structures.

Table 2.2: Examples of IEEE 802.15.4 radio transceivers and their consumption characteristics.

Part Number	TX 0dBm [mA]	RX [mA]
Testchip RF200 [c_2]	4.9	4.5
LTC5800-IPM SoC [94]	5.4	4.5
AT86RF231 [7]	11.6	10.3
CC2538 SoC [157]	24	20
CC2520 [156]	25.8	18.5
CC2420 [19]	19.5	21.8
MPR2400 [101]	17.4	19.7

fier (LNA). Once it receives a predefined sequence of bits, known as *preamble*, digital circuitry is triggered and the following bytes corresponding to the radio frame are stored in a local buffer. If the receiver is turned on, LNA amplifies the received signal even when there is no transmission on the air at all. This is called *idle listening* and it is a power-hungry operation that must be minimized to prolong device lifetime. Minimization of idle listening is the main function of radio duty-cycling protocols discussed in Chapter 6.

Many radio technologies exist, like IEEE 802.15.4, Bluetooth Low Energy (BTLE), IEEE 802.11, LoRA, and they differ in modulation schemes used, data rates, frame sizes, operating frequencies. We give examples of IEEE 802.15.4 radio transceiver hardware in Table 2.2. IEEE 802.15.4 specifies *both* the physical layer (modulation, data rate, frequency) and the Medium Access Control (MAC) protocol that mediates access to the common wireless medium of multiple nodes present in an IEEE 802.15.4 network. The MAC protocol of IEEE 802.15.4 standard was amended in 2012 by IEEE 802.15.4e [63] to increase reliability and robustness of wireless communication by supporting two channel hopping mechanisms: Time-Slotted Channel Hopping (TSCH) and Deterministic and Synchronous Multichannel Extension (DSME).

If we compare current draw values in Table 2.2 with those in Table 2.1, we can observe an important characteristics of IoT devices: energy consumption is dominated by radio usage. This is the key concept to keep in mind for any IoT protocol design, including security and the leitmotif of this thesis – it is beneficial to trade off radio exchanges for local computation.

Energy Source

Smart devices are used in various application contexts and it is typically the use case that determines the main source of energy. The predominant scenario is that of

a battery, but the IoT scale makes the daily battery recharge unfeasible, as we are used to with smartphones or laptop computers. For that reason, we commonly find devices powered by a pair of non-rechargeable AA batteries, each typically carrying a capacity of around 2200mAh. How soon the batteries will be depleted depends on the (average) current draw of the whole board, and we could see that radio transmissions and receptions play the dominant role. This introduces the notion of device *lifetime*, since the economical costs of battery replacement on hundreds or thousands of devices are often prohibitive.

Some systems employ energy scavenging techniques, like solar cells, piezoelectric elements, temperature gradients. We show some typical figures on achievable power densities in Table 2.3.

Table 2.3: Typical power density of different harvesting approaches [133].

Harvester	**Power Density**
Solar Cell	15 mW/cm^2
Piezoelectric	0.330 mW/cm^3
Vibration	0.115 mW/cm^3
Thermoelectric	0.040 mW/cm^3

The harvested energy can either be used only when available, or accumulated in the energy buffer – a rechargeable battery or a super-capacitor. Systems without any storage are small, unidirectional devices such as piezoelectric switches or impractical systems requiring large solar panels [133]. Medium term research projects aim at designing such systems that are feasible, practical and miniature in size by integrating all the necessary components on a single chip [121]. Today, we most commonly find energy-harvested systems that employ the energy buffer, whose capacity is dependent on the envisioned application.

Finally, some use cases allow IoT devices to be *mains-powered*, i.e. attached to the electric grid. Some examples are smart electricity meters, or smart light bulbs. Technically this requires the device packaging to integrate an AC-DC converter. Although in such cases energy is unlimited locally, we should always keep into account the IoT scale and the requirements that billions of such devices put on global energy production.

In conclusion, the amount of available energy per device in all cases stays very low which mandates the development of very efficient hardware and software techniques.

2.3 The GreenNet project

This thesis is part of the GreenNet project at STMicroelectronics. GreenNet project was launched in 2011 with the goal of designing and manufacturing a self-powered IoT device. The selected energy harvester for GreenNet boards was a

Figure 2.1: GreenNet board.

Photovoltaic (PV) system, i.e. solar panel, optimized for indoor energy harvesting. GreenNet targets IoT applications like home and building automation. We show in Fig. 2.1 a photograph of a GreenNet board.

Main focus of GreenNet design was low power. Boards were designed to operate in extremely low light conditions, as low as 150 lux. To put this number in context, light intensity just bellow a typical fluorescent lamp commonly found in office space is about 8000 lux [133]. At a distance less than 1.5 meters from the lamp, light intensity decreases to 600 lux. On the surface of an office desk, 2 meters away from the lamp, one can obtain around 300 lux, while surfaces bellow the desk get around 150 lux. The solar panel used on GreenNet boards (50 $\times$ 48mm), in these conditions can harvest from 1mA to 0.020mA of current [133]. In order for GreenNet boards to operate sustainably, internal hardware components must have extremely low power consumption. We depict the layout of GreenNet boards in Fig. 2.2.

Heart of GreenNet boards is STM32L1 [148] microcontroller based on ARM Cortex-M3 core that embeds many peripherals such as ADC, Digital to Analog Converter (DAC), several timers and comparators, and also the crypto acceleration core. It controls the external peripherals such as various sensors and Light Emitting Diodes (LEDs) (see Fig. 2.2). Prototype GreenNet boards also integrate an additional STM32F1 MCU whose sole purpose is flashing and debugging of STM32L1 without the need for an external debugger.

GreenNet team designed a Testchip RF200 radio transceiver, compatible with IEEE 802.15.4 standard and with best-in-class power consumption (see Table 2.2). Low power characteristics of Testchip RF200 are crucial for meeting stringent energy requirements of GreenNet.

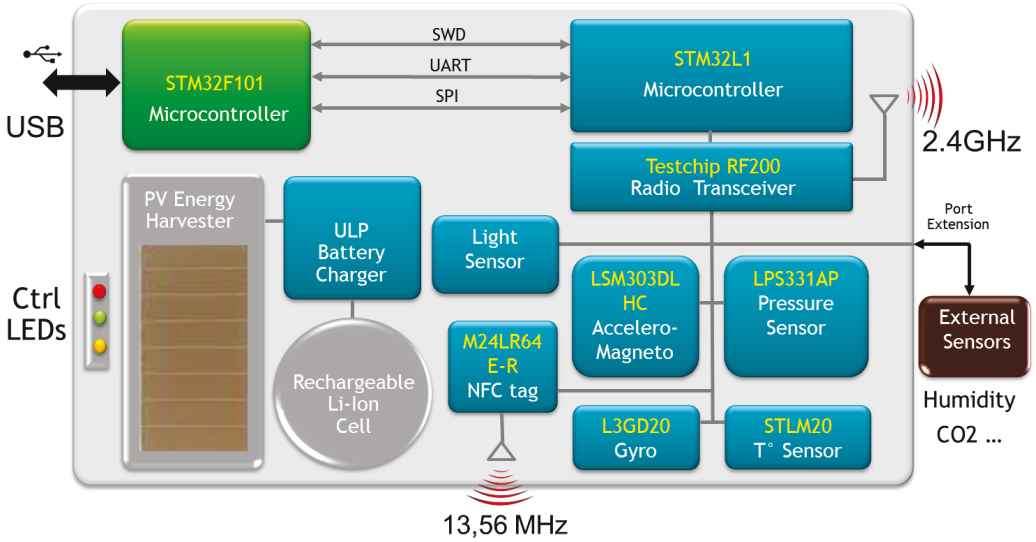


Figure 2.2: GreenNet board layout.

A small coin-cell Lithium-Ion rechargeable battery provides energy supply to the whole board. Capacity of this battery is only 25mAh but it provides a healthy compromise among a number of factors such as number of recharges, leakage, maximum energy capacity loss if the battery is repeatedly recharged after a partial discharge. For comparison, one AA battery holds 2200mAh.

GreenNet team also designed a Power Management Unit (PMU) that recharges the battery either from the solar panel or by leveraging the Universal Serial Bus (USB) connection during development [159, 160].

The Near Field Communication (NFC) transceiver available on GreenNet boards is used for device bootstrapping. For example, initial security keying material and network parameters can be communicated to the device in order to facilitate the joining of an IEEE 802.15.4 network.

2.4 Conclusion

We could notice that IoT devices pose interesting technical challenges to IoT system designers. Their processing capabilities are much lower than what we are used to with traditional computing platforms. In addition, both data and program memory are limited in order to keep the per-unit cost low. Because majority of devices are energy constrained, they need to sleep most of the time to preserve energy. Energy spent while devices are sleeping is due to the sleep mode leakages. These leakages are directly related to the size of data memory that needs to be retained while the device is sleeping. Therefore, future increases in memory for IoT microcontrollers are dependent on the advances related to leakage currents.

When the device is not sleeping, we could notice that radio receptions or transmissions dominate the energy consumption over local processing. Therefore, system

design and network (security) protocols should prioritize local computations over radio communication.

In the following chapter, we describe the communication stack and different protocols that allow IoT devices such as GreenNet boards to form self-sustainable networks and facilitate their integration with IoT. For a detailed overview of GreenNet optimizations related to energy harvesting, the reader should refer to Varga *et al.* [c_2].

Chapter 3

Standards-based Protocol Stack for Interoperatiblity

Efficient hardware is a prerequisite for low-power boards, such as GreenNet, to participate in the Internet of Things (IoT). Networking protocols optimized for low-power operation [133] [c_2] complement bare metal and allow the formation of networks that can meet application requirements.

In this chapter, we overview the communication stack that facilitates the integration of Wireless Sensor Networks (WSNs) with the IoT. This communication stack stems from the IEEE 802.15.4 radio transceiver and builds upwards using protocols standardized by Internet Engineering Task Force (IETF).

The chapter is organized as follows. In Section 3.1, we review the traditional abstraction layers of the Internet and discuss the repercussions of the separation-of-concerns concept on IoT system design. In Sections 3.2, 3.3, 3.4 and 3.5, we proceed by overviewing bottom-up the existing IoT solutions at each layer of the protocol stack. In Section 3.6, we brief the implementation choices of the GreenNet project. We conclude the chapter in Section 3.7.

3.1 Independent Layers and a Limited Energy Supply

Since the dawn of the Internet, we have dealt with complexity of computer networks through abstraction layers. Each layer has a specific service that it offers to the layer above, and builds upon the services provided by the layer directly below. Such a separation allows clean design of networking protocols and products that can interoperate at different layers of the protocol stack. We follow the traditional Internet protocol suite that consists of 5 layers: physical, link, network, transport and application layers.

Physical layer. The fundamental abstraction is the physical layer that is in charge of transmitting bits over the wire, air or any other transmission medium. Physical-layer protocols specify how signaling between the transceiver and receiver takes place and how upper-layer data units are encapsulated for physical transmission. For instance, IEEE 802.15.4 PHY specifies the format of the preamble used for synchronizing the two transceivers and that the first "data" byte corresponds to the length of the link-layer Protocol Data Unit (PDU). Physical-layer specifications detail the technical aspects of the transceiver, such as modulation rates, frequency bands, expected physical-layer timings and similar.

Link layer. Physical transmission and reception are services offered to the link layer which groups individual information bits into *frames*, the atomic unit of transmission over a given link technology. Link layer is in charge of "moving" frames from one node to the other. For wireless technologies, like IEEE 802.15.4, link layer typically provides integrity protection and reliability. Integrity protection detects any errors that may have occurred during reception by using error detection techniques, such as Cyclic Redundancy Check (CRC). Once the receiver is certain that the received frame does not contain any errors, for reliability, it can signal it to the transmitter by sending an Acknowledgment (ACK) frame. Because wireless medium is broadcast in nature, a Medium Access Control (MAC) protocol needs to govern communication among multiple nodes and avoid the situations when multiple nodes speak at the same time – *collisions*. Collisions are costly as they incapacitate the receiver from decoding the frame correctly. Collisions lead to retransmissions which increase delays and energy consumption. Another important role of the link layer is to preserve energy – we saw in Chapter 2 how expensive it is to keep the radio transceiver on and perform idle listening. Link-layer protocols in energy-constrained environments switch the radio transceiver off as much as possible – this operation is called Radio Duty-Cycling (RDC). When a transmitter decides to send a frame over the air, the destination node needs to be awake and listening. Otherwise, the intended receiver will not have received the frame which will cause the transmitter to retransmit and waste energy. Concepts of RDC are tightly coupled with MAC and considering one without the other often leads to severely degraded performance.

Network layer. Since the link layer handles communication of two nodes directly connected to a given physical link, network layer abstracts various link technologies and allows interconnection of heterogenous networks. Network layer provides the abstraction of an endpoint – a network node reachable potentially multiple link-layer hops away. Each of the *hops* on this path can be accessible over a different link-layer technology, although we will mostly consider scenarios where packets are forwarded multiple hops over IEEE 802.15.4. There are two fundamental tasks of the network layer: addressing and routing. Addressing, as each host needs to be (uniquely) addressable in the global network and routing because packets need to find their route from the sender to the destination node, over multiple hops. Internet Protocol (IP) is the glue that binds the Internet together [84] and inevitably makes part of the IoT.

Transport layer. Because every router on the path examines network-layer headers, network layer and IP do not provide any guarantees on reliable or orderly delivery of packets exchanged between two endpoints. Although different link-layer protocols provide reliability, this does not imply reliability between two endpoints, multiple hops away, as packets can be dropped locally, for instance due to limited buffer sizes. Another important service that is offered by the transport layer is multiplexing of different applications that can run on a host by port number signaling.

Application layer. The layered approach reduces the complexity of design and analysis of different problems that arise while interconnecting heterogeneous sys-

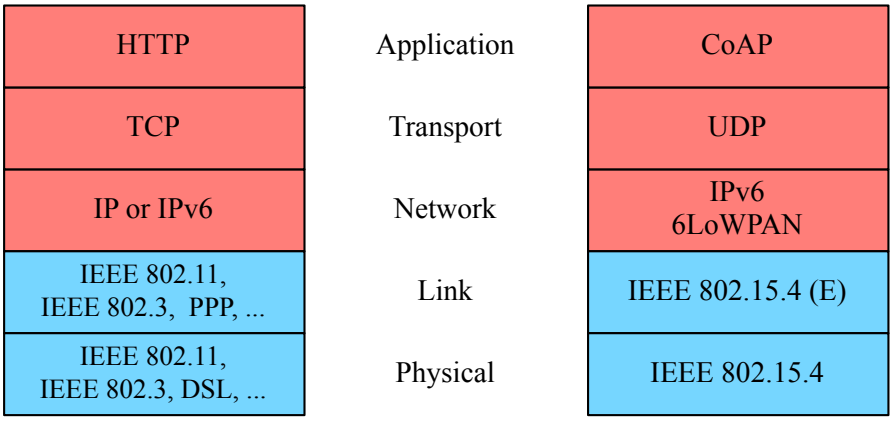


Figure 3.1: Protocol suites typically used in (a) traditional Internet, and in (b) IP-based Wireless Sensor Networks.

tems. Moreover, the layered approach allows the application developer to be unfamiliar with the problematics at different layers. The developer uses the Application Programming Interface (API) to exchange information between its applications running on different hosts and does not need to worry about routing issues, wireless channel characteristics and similar. Application-layer protocol is in charge of transferring information needed by the actual application – be it a web page, or a temperature reading from a smart house. The *separation of concerns* provided by the layered approach has indeed enabled a wide range of developers to contribute to the interconnected world, as we know it today.

In Fig. 3.1 we depict the specific protocols as typically used in the Internet (Web) and the one considered in this thesis for the integration of WSNs with IoT. Before we overview different choices, it is important to stress an important difference between traditional Internet hosts and our constrained devices – energy supply. Traditional Internet hosts are either mains-powered or have batteries that are constantly recharged during their use (smart phones, laptops). Constrained devices are expected to operate on a single battery for years or to be self-sustainable from the harvested energy. This imposes a strict requirement on any IoT product, that at times leads to violations of layer purity, as seen in the Internet. Being *efficient* is paramount.

Efficiency leads to vertical integration and cross-layer interactions that aim to optimize the stack as a whole. In practice, this means that the application often needs to be aware of the underlying RDC schedule or a routing protocol that is tightly integrated with the link-layer technology. Radio Duty-Cycling is often not just the duty cycling of the radio. More often, it is the duty cycling of the whole board, with both Microcontroller Unit (MCU) and radio in sleep mode and all but most-necessary sensors switched off. There is a single and limited energy supply for all the layers, after all.

3.2 IEEE 802.15.4 Physical Layer

We begin our quest up the protocol stack in Fig. 3.1 from the physical layer (PHY). IEEE 802.15.4 [62] is arguably the most prominent standard in low-power technology and the one that was chosen for GreenNet project. IEEE 802.15.4 standard specifies multiple physical layers that can be used in different parts of the world, depending on local regulations. Our focus is on the physical layer in the Industrial, Scientific, Medical (ISM) band at 2.4GHz that guarantees world-wide use free of any licensing requirements, that is in practice the most widely deployed. The 2.405GHz - 2.480GHz band is split into 16 frequency channels that are 5MHz apart. Each channel is only 2MHz wide, with the remaining band used as a guard against adjacent-channel interference. The bands at 868MHz and 915MHz with better propagation characteristics are also popular but they are only available in certain geographical regions and have a single frequency channel available.

IEEE 802.15.4 physical layer at 2.4GHz uses Direct Sequence Spread Spectrum (DSSS) technique for robustness: each 4 bits of data are encoded as 32 chips (physical bits) [115]. This helps recover from errors caused by narrow band interference. Offset Quadrature Phase-Shift Keying (O-QPSK) modulation is then used and results in physical rate of 2 Mchips/s and effective data rate of 250 kb/s.

Before the upper-layer information can be exchanged, transmitter starts by sending a *preamble*, a pre-defined sequence of ones and zeros that allows the receiver to synchronize. Transmission of the preamble lasts 128µs, and is followed by a Start of Frame Delimiter (SFD), another pre-defined sequence. SFD signals that the subsequent byte corresponds to the physical-layer payload. First byte of the payload indicates the length of the encapsulated radio frame. IEEE 802.15.4 specifies that the maximum length frame, i.e. link-layer Maximum Transmission Unit (MTU), can be 127 bytes, which is commonly reflected in radio buffers [19].

3.3 Medium Access Control and Radio Duty-Cycling

Where the energy is not an issue, radio can continuously listen to the channel and wait for a transmission. The problem in such wireless environments is mostly how to access the common medium and how to maximize the throughput and fairness among all the nodes in the network. This is typically handled using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol, as in IEEE 802.11 networks. We will refer to this mode of operation as *always on*.

In our case, consumption is dominated by radio usage (see Chapter 2) and particularly idle listening. Duty cycling is therefore a cornerstone technique for achieving long lifetimes of IoT devices. A typical IoT node with an IEEE 802.15.4 radio will deplete a 2200mAh AA battery in about a week, if the radio is left on continuously (either receiving or transmitting). State-of-the-art duty cycling protocols reduce the duty cycle below 1%, thereby extending the device lifetime to several years. The price of such aggressive duty cycling is an increased network delay and reduced throughput.

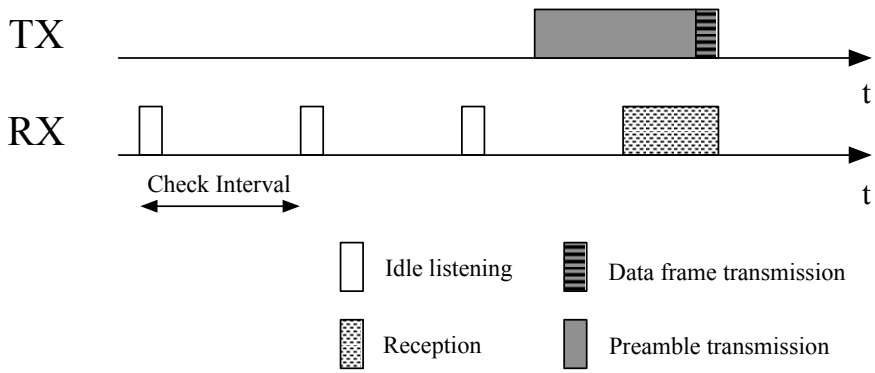


Figure 3.2: Basic principle of preamble sampling techniques.

RDC techniques are tightly bound to the channel access method. In the following, we overview RDC and MAC schemes with the most relevance for this thesis, that are either a part of IEEE 802.15.4 standard or can leverage its frame format.

3.3.1 Preamble Sampling Techniques

The MAC part of IEEE 802.15.4 standard [62] defines two operating modes. In the *non-beacon* mode, all nodes use CSMA-CA for channel access with contention, which implies that they should be always on to avoid deafness. Fortunately, the research community had taken over and many schemes were proposed to complement the non-beacon mode of the standard. We focus on preamble sampling techniques, arguably the most popular method that does not require network-wise synchronization.

We depict in Fig. 3.2 the main principles of preamble sampling. The idea is to precede every transmission with a long preamble. Although this preamble could be implemented as a physical-layer preamble, it is more often implemented as a sequence of radio frames with a pre-defined structure, called *strobes*, which avoids hardware-level modifications. Receiver periodically wakes up and verifies if it can detect a strobe on the channel. We refer to the wake-up period as Check Interval (CI) and a typical value is 125ms. The lower the `CI`, the more often nodes check the medium, and the higher their idle radio duty-cycle. For this to work, the transmitter needs to emit the preamble for at least the `CI`. If the receiver detects the preamble, it stays awake to receive the data frame that follows.

Many derivations exist [17, 33, 38, 9, 99] but we only attempt to present general principles. X-MAC [17] adds the receiver address in each *strobe*, so only the destination node keeps its radio on to receive the data. The duration between two strobe transmissions is enough for the recipient to send a short ACK frame, letting the transmitter know that it has woken up and is ready to receive the data frame. Upon the reception of ACK frame, the transmitter halts sending new *strobes* and instead, it transmits the data packet. Another protocol called ContikiMAC [33] operates similarly and transmits the actual data frame multiple times until an ACK is received.

Nodes contend for the access to the wireless medium in a traditional manner – using CSMA/CA. Operation in preamble sampling protocols effectively *emulates* the always on mode of operation and presents a clean binding with the upper layer – RDC protocol does not introduce any dependency on network topology. As such, mesh networks (peer-to-peer) can be created using routing protocols at the network layer. Most notably, preamble sampling approaches do not require the nodes in the network to (re)synchronize their clocks.

There are several inhibitors to the wide adoption of preamble sampling RDC schemes. They are not standardized and a product adhering to X-MAC would not interoperate with another scheme. Broadcast transmissions can be prohibitively expensive for the transmitter because there are no acknowledgment frames and the solution is to transmit the maximum length preamble (or maximum number of strobes). Another disadvantage is that all nodes in the network must use the same `CI` value, which does not allow energy-wise heterogeneous networks to operate in a deterministic way. To illustrate, one can imagine a network composed of battery-operated and energy-harvested nodes. A lower bound on duty-cycle imposed by a given `CI` value may be acceptable for battery-operated nodes but not their harvested counterparts. Moreover, setting `CI` to a large value to reduce the lower bound of RDC at the receiver imposes a significant burden on the transmitter. For that reason, most commonly found `CI` is 125ms, which provides a healthy tradeoff between the transmitter load and the achievable RDC of the receivers. Finally, depending on the application requirements, duty cycling is often done for the whole board, not just the radio. RDC based on preamble sampling would mandate the whole board to be woken up with a fairly short interval[1] when all that application needs is, for example, one measurement every 4 minutes.

3.3.2 Beacon-enabled IEEE 802.15.4

The *beacon-enabled* mode of IEEE 802.15.4 aims at saving energy: nodes are synchronized with periodic beacons and only wake up at specific instants to communicate [c_2]. Nodes in this mode have two different roles:

- *Coordinators*: they send beacons to delimit time intervals called superframes. A beacon invites the devices associated with a given coordinator to send their frames during the Contention Access Period (CAP).
- *Devices*: they are leaves in the tree-based topology and communicate only with their coordinators.

A network node is configured either as a coordinator or a device, or both when it forwards traffic. Fig. 3.3 presents the superframe structure used in the *beacon-enabled* mode. Coordinators transmit beacons every Beacon Interval (BI) while a

[1] In general, we cannot assume that the radio hardware can wake up on its own to follow the RDC schedule. This is generally managed by the MCU although the benefits of having a dedicated hardware module for such a task are evident and present in some commercial products.

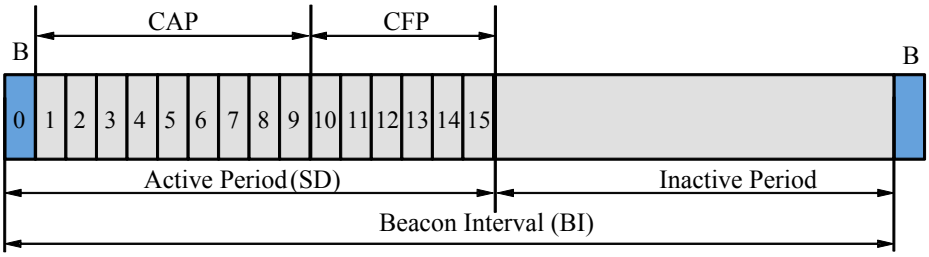


Figure 3.3: 802.15.4 superframe structure. GreenNet nodes only use the Contention Access Period (they do not use the Contention Free Period).

superframe lasts for a Superframe Duration (SD). The intervals depend on the corresponding Beacon Order (BO) and Superframe Order (SO) parameters:

$$\text{BI} = aBaseSuperFrameDuration * 2^{\text{BO}} \tag{3.1}$$

and

$$\text{SD} = aBaseSuperFrameDuration * 2^{\text{SO}} \tag{3.2}$$

where ($0 \leq \text{SO} \leq \text{BO} \leq 14$, *i.e.* $15.36ms \leq \text{SD}, \text{BI} \leq 4.2\ min$). Nodes may sleep during the inactive period of the superframe.

3.3.2.1 Topology in Beacon-Enabled Mode

An IEEE 802.15.4 network in beacon-enabled mode can have a star or cluster-tree topology. Star topologies at 2.4GHz are often not practical due to the limited range where nodes can be deployed. In the cluster-tree topology (see Fig. 3.4), the Personal Area Network (PAN) coordinator is the root of the multi-hop network. It serves as a data sink and represents the first coordinator in the cluster-tree. Nodes are unassociated at the beginning and they wait for beacons (passive scanning), even on several channels, to join the network.

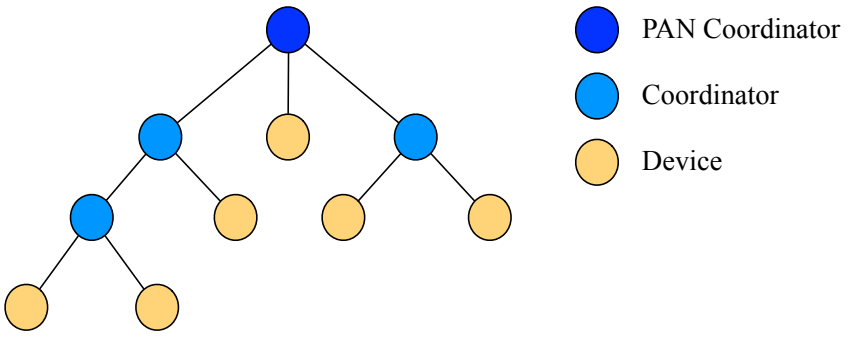


Figure 3.4: Example cluster-tree topology in beacon-enabled IEEE 802.15.4.

When a node receives a beacon from a neighbor, it may associate with it by exchanging control frames. A coordinator maintains a list of devices and responds with an association response if it has not reached the maximum limit of associated

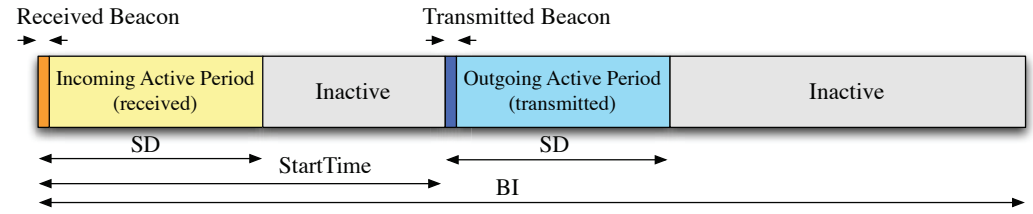


Figure 3.5: Incoming and Outgoing superframe structure in IEEE 802.15.4.

devices. After association, the node may become a coordinator itself: it periodically sends its beacons to invite other nodes to associate.

Beacons also indicate the initial instant of the Active Period and contain the list of destination addresses for frames stored at the coordinator. During the Active Period, a device either retrieves frames by transmitting a `data-request` frame if its address was present in the pending destination list or transmits its `data` frames to the coordinator. Note that the coordinator never initiates a transmission, but only replies to solicitations from its devices. Devices have to explicitly request their frames from a coordinator, which enables switching off their radio and saving energy without deafness. This also allows devices to sleep extensive periods of time, as they are not obliged to wake up for every beacon [c_2]. For instance, GreenNet temperature sensor wakes up only once every 4 minutes. To avoid collisions, all devices use the slotted CSMA/CA method to access the medium during CAP.

Coordinators act as devices with respect to other coordinators when they forward packets to the root of the cluster-tree topology. To support forwarding over multiple hops, IEEE 802.15.4 defines the Outgoing (maintained by a coordinator on the path to the root) and the Incoming superframes (maintained by the node for receiving frames from its devices) (see Fig. 3.5).

Beacon-enabled mode presents several advantages over preamble sampling techniques. It is standardized, and it allows devices with heterogeneous energy requirements to form a network.

It does so at the cost of network-wise synchronization by using beacons and by forcing a specific link-layer topology. The IEEE 802.15.4 standard does not specify the details of the cluster-tree construction algorithm leaving its implementation open. ZigBee [185] defines a protocol for cluster-tree construction based on three constraints: a maximum number of devices, a maximum number of coordinators and a maximum depth. If considered independently, the link-layer topology "forces" the routing protocol to use paths that may not be optimal. For instance, if the cluster-tree is constructed with the goal of minimizing the number of hops to the root, paths in the network may not be optimal in terms of link quality. For this reason, in Appendix B we study how the two processes – construction of the cluster-tree and of the routing paths – can be merged.

Beacon-enabled mode of IEEE 802.15.4 is the principal energy-saving technique for GreenNet. For more details on the specific mechanisms developed in the scope

of the GreenNet project, the reader should refer to Varga *et al.* [c_2].

3.3.3 Time-Slotted Channel Hopping

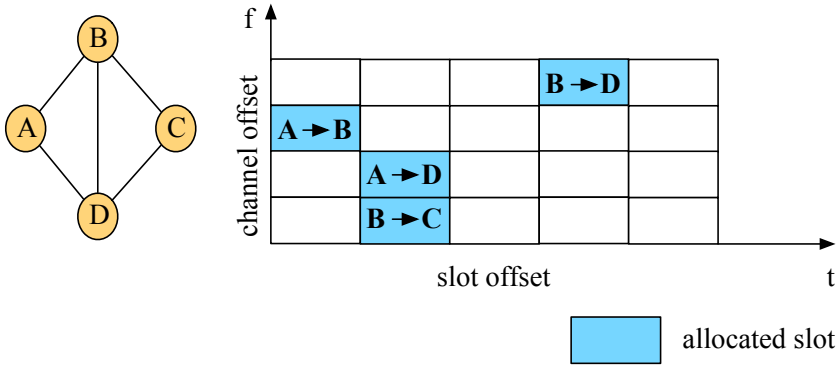


Figure 3.6: Example mesh topology and TSCH schedule.

Both the *non-beacon* and *beacon-enabled* modes of IEEE 802.15.4 consider network operation at a single frequency channel. Watteyne *et al.* showed [175, 176] that external interference and multi-path fading severely degrade the quality of a wireless link at 2.4GHz, both in indoor and outdoor deployments. As an extension to IEEE 802.15.4-2011, the IEEE 802.15.4e-2012 standard [63] defines the Time-Slotted Channel Hopping (TSCH) mode, which uses "channel hopping" to combat external interference and multi-path fading.

In a TSCH network, time is cut into timeslots, each long enough for a transmitter to send a `data` frame to a receiver, and for the receiver to send back an `ACK` indicating successful reception. Duration of a timeslot is constant for the whole network (typical value is 10ms) and timings within the timeslot are precisely defined. L successive timeslots are grouped into a "slotframe", which continuously repeats over time. A communication schedule indicates to each node, for each slot in the slotframe, what to do (transmit, receive or sleep) and on which channel offset. We depict a simple example mesh topology in Fig. 3.6.

The scheduler provides the nodes with a channel offset corresponding to a given communication slot. Nodes uniformly circulate over n_{ch} available physical channels, knowing Absolute Slot Number (ASN), the number of slots that elapsed since the network was deployed and the channel offset in a given slot [115]. Physical frequency f where a transmission occurs is calculated as:

$$f = F[(ASN + channelOffset) \mod n_{ch}], \tag{3.3}$$

where F is a lookup table that matches channel to a physical frequency. In the 2.4GHz band, n_{ch} is typically 16 but certain channels can be blacklisted if, for example it is known that there is a strong interferer present. If the length of the slotframe L is a relatively prime number, Eq. 3.3 ensures that a communication slot in the schedule rotates over the available channels in successive slotframes.

The communication schedule can be built in a centralized or distributed fashion. The scheduler (either a centralized computer or a distributed protocol) builds and maintains the schedule in order to match the link-layer resources (timeslots scheduled between neighbor nodes) to the applications needs (number of packets per second, latency requirements). In the centralized case, each node reports to the scheduler its radio neighbors and the estimated link quality, which serves as an input to the scheduling algorithm. Scheduler has precise control over throughput, latency and energy consumption tradeoffs in the network, with the granularity of a single node.

In TSCH networks, beacons are not necessary for synchronization purposes[2]. Nodes keep synchronization with the network by using regular `data` frames. `ACK` frames feedback how early or late the sender is in respect to the ideal boundary when the packet should have been transmitted, from the perspective of the receiver. Each node in the network has its *time parent* and uses the feedback in `ACK` frames to correct its clock. When the *time parent* corresponds to the default route on the path to the root, synchronization is maintained by using regular, convergecast application traffic with no additional overhead. When a time parent initiates communication with its child, the child can correct its clock by estimating how early or late the `data` frame was sent. Nodes need to exchange packets to stay synchronized or they will drift apart, as each node's crystal oscillator beats at a slightly different frequency. This puts a theoretical upper bound on the time without any communication. A typical value with $\pm 15 ppm$ accuracy and 1ms guard times is around 30s. When the application traffic is absent, a node can resynchronize to its time parent by sending a short *keep-alive* packet, without any payload.

Main advantages of TSCH are efficiency due to tight synchronization within a timeslot, and reliability due to channel hopping. Unlike *beacon-enabled* mode, TSCH allows the formation of a full mesh network and therefore provides means to the upper-layer routing protocol to leverage path redundancy in the mesh. The exact topology that is exposed to the upper layer depends on the schedule. TSCH naturally supports energy-wise heterogenous networks and the most constrained nodes can sleep in all but a couple of slots during the slotframe.

The requirement to keep tight synchronization within one timeslot is in the same time the biggest disadvantage of TSCH. To achieve sleep times on the order of 4 minutes, as it was demonstrated with GreenNet and *beacon-enabled* mode, it would be necessary to employ drift estimation and compensation techniques [23] that have dependency on environmental conditions and hardware aging. TSCH withstands as the most widely deployed and field-proven WSN technology that has transitioned IoT from an academic concept to reality [115].

[2]Beacons in TSCH mode may be used only during the initial network formation to advertise the presence of the network. Once the network is formed, beacons are no longer necessary and their transmissions can be switched off.

3.4 Internet Protocol and Routing in a Mesh

The limited address space of Internet Protocol version 4 (IPv4) has long been an issue but the transition to Internet Protocol version 6 (IPv6) has been slow in practice. In that sense, IoT can be regarded as an opportunity for a fresh start. IoT devices are expected to and currently do run the version 6 of the infamous Internet Protocol.

Addressing. An important piece of the IPv6 addressing structure in the IoT context is the 64-bit Extended Unique Identifier (EUI-64), imprinted by a manufacturer for each communication interface. IPv6 in combination with EUI-64 allows the device to self-assign its IP address, eliminating the need of external configuration by a protocol or manually by an administrator. When the IoT device first joins a new network, it learns the global IPv6 prefix that it uses in combination with EUI-64 to construct a globally reachable IPv6 address. Deployments that we witness today commonly intersect communication from the outside at the network gateway, for security reasons due to the energy constraints. Devices are still uniquely addressable among each other, within the local area network.

IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) adaptation layer. IPv6 header contains 40 bytes and it requires an MTU of 1280 bytes. When IPv6 is coupled with IEEE 802.15.4 frame sizes of 127 bytes, the need for an adaptation layer is evident. 6LoWPAN [108] is a shim adaptation layer in-between IP and IEEE 802.15.4. 6LoWPAN defines a mechanism for fragmentation and re-assembly of IP packets carried in IEEE 802.15.4 frames. However, fragmentation is *undesirable* due to severe performance issues with lossy wireless links. For that reason, 6LoWPAN also specifies a compression mechanism, in order to reduce the overhead of IP header. 6LoWPAN compression leverages shared state of all devices in a local network (such as network prefix) or omits the fields that can be inferred from other layers. Typically, this results in available application-level payload size of around 80 bytes. For a detailed overview of 6LoWPAN compression techniques, the reader should refer to RFC 4944 [108] and the subsequent updates.

Routing in a Mesh. Each node in a mesh is a router and can forward packets originated at other nodes in the network towards their destination. Routing can be performed either at the link layer or at the network layer. In the former case, it is denoted as *mesh-under* and in the latter as *route-over*. Mesh-under schemes, forward independently each radio frame and the IP-layer sees the whole local network as a single hop [97]. This creates an undesirable effect with broadcast frames – each broadcast at the IP level results in flooding over multiple hops at the link layer – and prevents the design of upper-layer protocols that leverage a given physical topology. In route-over schemes, IP header is examined at each hop on the path, which also necessitates 6LoWPAN decompression, but allows multiple backhaul-interconnected constrained networks to be a part of the same routing architecture. To forward packets along a route, each node maintains a *routing table* that indicates the next hop for a given destination address. As the convergecast traffic is dominant in WSNs, the routing protocols typically build a tree, rooted at the PAN coordinator.

We overview the principle routing protocol for IP-based WSNs in Appendix A and study its interaction with beacon-enabled mode of IEEE 802.15.4 in Appendix B.

3.5 Transport and Application Layers

Internet hosts run multiple applications and the transport layer multiplexes among them by using the concept of a port number. As the network-layer IP protocol does not provide end-to-end reliability, Internet applications use Transmission Control Protocol (TCP) at the transport layer that establishes a reliable channel between two endpoints. The reliability and orderly delivery with TCP are achieved with transport-layer sequence numbers and end-to-end retransmissions when a lost packet is detected. Energy-wise, this has proven to be very costly in constrained networks as end-to-end retransmissions traverse the whole network and affect each hop on the path [115]. The protocol stack in Fig. 3.1 reflects this, and the lightweight User Datagram Protocol (UDP) is used instead of TCP, bearing no transport-layer state between two endpoints. UDP header traditionally caries 8 bytes of overhead for port number signaling, transport-layer length and checksum bytes but 6LoWPAN additionally allows UDP compression leveraging inter-layer dependencies. UDP, however, does not provide any guarantees on reliability or orderly delivery, which is left to the application.

Application-layer protocols enable interoperability between different applications and provide application-independent semantics that facilitates content representation [115]. The success of the Internet is much due to the success of Hypertext Transfer Protocol (HTTP) and the Web that enabled unprecedented information sharing. Benefits of integrating IoT data from ubiquitous sensors and actuators in the same architecture are evident.

HTTP builds upon a Representational State Transfer (REST) architectural style that provides properties such as scalability, performance or reliability, suitable for the global scale of the Web [41]. Billions of IoT devices add unprecedented amounts of data to the existing system(s) so holding on to the REST properties is paramount. But one should in no circumstances forget the constraints that individual IoT devices inherit. HTTP is in that sense not well adapted to IoT devices – it expects a reliable TCP channel beneath, synchronous request-response communication between a client and a server, and adds considerable packet overhead, as it is not concerned with payload sizes.

3.5.1 Constrained Application Protocol (CoAP)

IETF Constrained RESTful Environments (CORE) working group standardized CoAP [144] to answer the needs of IoT devices. CoAP is not a blind compression of HTTP, although it is commonly referred to as such due to the same RESTful design and purpose. Instead, CoAP implements a subset of HTTP features relevant to IoT devices and further specializes for constrained environments by leveraging typical traffic patterns and by minimizing overhead.

The most notable features of CoAP are [144]:

- UDP transport with optional reliability, supporting unicast and multicast requests.
- Asynchronous message exchanges.
- Low header overhead and parsing complexity.
- Simple proxy and caching capabilities.
- A stateless HTTP mapping allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP interfaces to be realized alternatively over CoAP.

CoAP leverages the traditional client-server architecture where a client interested in a given resource makes a *request* towards the server, hosting that resource. The server prepares a *response* that potentially encapsulates a *resource representation* corresponding to the requested resource. For example, a request to resource "temperature" would result in response containing temperature measurement in the payload. There could also be a request to change the value of some resource on the server, in which case the response typically acknowledges the change. The separation of client and server roles in IoT environments is not as straightforward as in the traditional Internet. The same device often executes both roles, depending on the application semantics. The term "server" should by no means be associated with powerful machines hosting HTTP servers in the Internet, as in our context CoAP "server" corresponds to a constrained IoT device exposing its resources to CoAP clients (e.g. gateways, smart phones, or other IoT devices).

Resources in CoAP are, analogously to HTTP, identified with a Uniform Resource Identifier (URI) [14], and are organized in a hierarchical manner:

coap-URI = "coap:" "//" host [":" port] path-abempty ["?" query] .

CoAP separates *message* and *request-response* semantics. A single request may correspond to multiple exchanged messages, for example when the server asynchronously notifies the client of resource changes. Server may also not be able to respond with a resource representation immediately, in which case it indicates to the client that it received the request and that it will respond at a later point in time – the so-called *separate response*. Requests and responses share the common message format, depicted in Fig. 3.7.

Each message carries an identifier, Message ID, that is used for duplicate detection and optional reliability. This functionality of CoAP effectively complements UDP and leaves to the application to decide which messages, if lost, should trigger expensive end-to-end retransmissions. Clients can mark a message as *Confirmable* or *Non-Confirmable* by setting the appropriate Type (T) field. The corresponding response can be marked as *Acknowledgment* or *Reset*, depending on the processing outcome. If no response is received to a *Confirmable* message after a timeout

(default value is 2s), the client starts the retransmission phase with exponential backoff.

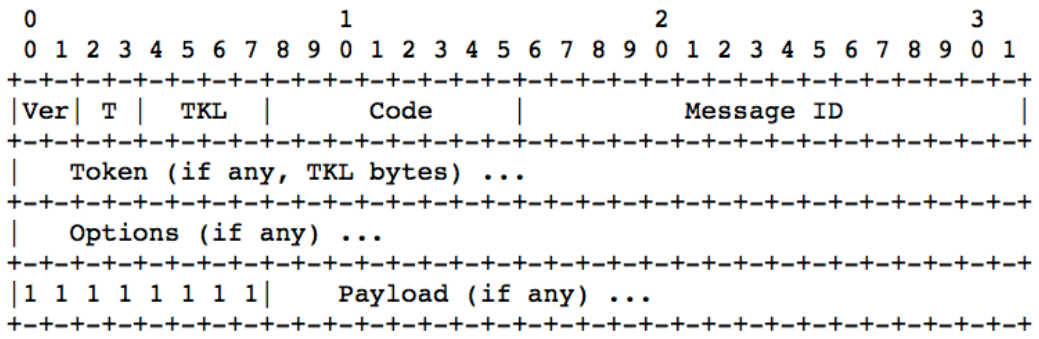


Figure 3.7: CoAP message format [144].

A response is matched to a request by using an optional identifier, called token (see Fig. 3.7). Code field signals whether a message is a request or response and the corresponding code. In the case it is a request, it encodes the request method, while in the case of response it encode the response code. Similarly to HTTP, response codes starting with "2" indicates success, "4" indicates client error, and "5" signals server-side error. Variable number of CoAP options follow the header and token. Options are Type-Length-Value (TLV) encoded and can carry URI, signal content format in the payload, maximum time a response may be cached before it is considered not fresh, and similar.

CoAP implements four REST methods:

- **GET** retrieves the resource representation that corresponds to the request URI. It is an idempotent (multiple invocations have the same effect) and safe (retrieval-only) operation.
- **POST** requests that the representation enclosed in the request be processed. The actual function performed by the POST method is determined by the origin server and dependent on the target resource. It usually results in a new resource being created or the target resource being updated [144]. It is neither safe nor idempotent.
- **PUT** updates the requested URI with the enclosed representation. It is not safe but is idempotent.
- **DELETE** requests the deletion of the resource identified by request URI. It is not safe but is idempotent.

Asynchronous exchanges and multicast. The dominant traffic pattern in WSNs is convergecast, where sensors send their readings towards an aggregation point, for example a gateway. It would be extremely inefficient if all the sensors in

the network would need to be queried to send each new reading. Moreover, applications that make use of presence sensors, acceleration alarms, or critical changes in sensor readings need to be informed *asynchronously*. CoAP handles this type of traffic with the *Observe* option [55] – a client makes a single GET request to the resource identified by request URI, but includes the Observe option which indicates to the server that it should notify the client whenever the state of the "observed" resource changes. Server then locally monitors the resource (e.g. temperature) and sends a response, matching the token in the original request, with a new value. This is an extremely important feature from the energy point of view, as the sensor can adapt the CoAP "notification" rate to its local energy conditions (e.g. intensity of harvested current). Finally, CoAP allows a client to make a request to an IP multicast group, which can minimize the number of messages exchanged in the constrained network. This can be very convenient for applications involving actuation. The typical example is simultaneous switching on or off the lights in a building [123].

Proxying and caching. A proxy is an application-level intermediary that stands between a client and a server and can perform requests on client's behalf. For instance, proxy can serve a request from a local cache and avoid the exchange with the energy-constrained CoAP server. Proxies are a fundamental component of REST architecture as they reduce traffic, response time and alleviate servers. A CoAP server can also maintain a local cache in order to avoid performing an expensive sensor reading[3] for each request. A server indicates the validity of a response by leveraging the *Max-Age* option, which signals the maximum time a response may be cached before it is considered not fresh [144]. CoAP distinguishes between two types of proxies:

- **Forward proxy**, that is explicitly selected by the client using Proxy-URI option in the request. Forward proxy can serve the requests from the local cache if they are still valid, or forward the request towards the CoAP server.

- **Reverse proxy**, transparent to the client that can for instance expose resources of the entire local area network, as if they were its own. Reverse proxy can be used on the network gateway to completely offload the constrained devices. For instance, gateway could act as a CoAP client towards the WSN and observe various resources, but offer the proxying functionality towards the Internet, serving responses from the local cache.

Mapping with HTTP. Since CoAP implements a subset of HTTP functionalities, there is a straightforward mapping between the two protocols. This can be implemented with a forward proxy that translates to HTTP when the message leaves the constrained network, and to CoAP when a message comes from the outside. For instance, a legacy device that only supports HTTP could talk to IoT

[3]For instance, measuring carbon-monoxide is a power consuming operation that can last several tens of milliseconds.

devices over such a proxy, and there would be no need for memory-limited devices to support complex HTTP semantics.

Security. CoAP mandates Datagram Transport Layer Security (DTLS) [129] as the default security solution. It is interesting to remark that *none* of the features discussed above behave satisfactorily when DTLS is considered in the picture:

- With asynchronous exchanges, DTLS necessitates keep-alive messages in the other direction, which questions the original purpose of asynchronous notifications.
- DTLS and multicast are inherently incompatible.
- DTLS cannot provide end-to-end security over proxies and the workaround in the Internet typically involves proxy server as a trusted intermediary or a tunneled connection, which may not be possible in IoT environments with CoAP.
- Mapping CoAP to HTTP and vise versa, cannot be performed when DTLS is used because integrity protection is performed at the transport layer.

3.6 GreenNet Implementation Choices

The principal communication stack used by GreenNet project is based on *beacon-enabed* IEEE 802.15.4 and is implemented as part of the Contiki Operating System (OS) for constrained devices [34]. Contiki has an event-based kernel and implements multithreading through *protothreads* [36], an efficient, stack-less construct, particularly suited for memory-constrained devices. The absence of a dedicated stack per protothread implies that local variables are not preserved over context switches. Contiki implements the discussed protocols and many RDC mechanisms. Contiki architecture is highly modular and allows customizations at each layer of the protocol stack. GreenNet complemented Contiki through the implementation of *beacon-enabled* mode of IEEE 802.15.4, Lightweight Routing Protocol (LRP) [85] developed specifically for the constraints of GreenNet, a multitude of cross-layer optimizations related to energy harvesting and *beacon-enabled* mode, and the standards-based security solutions presented in the remaining of this thesis [c_2].

In parallel to the principal GreenNet stack [c_2] and in order to support a complementary solution based on the prevailing TSCH protocol [115], we have extended OpenWSN open-source project [177] with support for GreenNet boards. Core of the OpenWSN project is an implementation of IEEE 802.15.4e that executes in the interrupt context and is driven by hardware events. Advantage of such approach is timing accuracy, that is mandatory for tight timing requirements of TSCH, as we will see in Part II.

3.7 Conclusion

We could notice that research efforts centered around constrained devices of the IoT have resulted in solutions at different layers of the protocol stack. Physical layer solutions, such as IEEE 802.15.4, provide requirements and guidelines on the design of low-power and low-cost radio transceivers. Link-layer schemes aim at minimizing the energy expenditure by reducing the idle listening of the radio transceiver. To integrate the IoT devices with the existing Internet infrastructure while keeping the overhead at minimal, standardization bodies defined the 6LoWPAN compression scheme. Recognizing the performance issues of end-to-end acknowledgments and the burden they put on constrained networks, IoT solutions typically use the UDP protocol, instead of TCP widely used in the traditional Internet. Finally, to account for the IoT communication paradigms such as asynchronous exchanges or group communication and the need for caching in order to hide the unavailable devices from the application developers, standardization bodies defined the CoAP protocol. The resulting IoT communication stack is quite different from the one we use on traditional computing platforms.

However, when it comes to security, we currently rely on the traditional Internet solution, DTLS, which results in application-level incompatibilities and forces the system designer to choose between product features and product security. We tackle these incompatibilities in Part II, and proceed by discussing the cryptographic primitives that are used on IoT devices and their performance.

Chapter 4

Cryptography and Constrained Devices

The main contributions of this thesis tackle different security aspects of the Internet of Things (IoT) communication stack. Whether the security issues are local to a single device, radio exchange of two network motes or more global like communication with a smart phone on the other side of the Globe, cryptography is used as a fundamental building block at multiple layers. In this chapter, we give a brief overview of cryptographic algorithms typically used on constrained devices and mandated by various IoT standards.

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication [103]. Many of these aspects and related techniques are fields on their own within the vast field of cryptography. Confidentiality is typically achieved with encryption schemes, data integrity with hash functions, entity and data origin authentication through keyed hash functions. However, in the context of constrained devices and due to the need to minimize the code size and thus the number of used algorithms, we tend to use the primitives that leverage the fundamental building block – a symmetric block cipher.

We devote Section 4.1 to background overview of symmetric-key primitives based on Advanced Encryption Standard (AES). We detail in Section 4.2 practical challenges of implementing some of these primitives on constrained hardware and discuss the contributed software architecture that aims at maximizing the performance and reducing the development time when implementing cryptographic support on a new IoT device. In Section 4.3, we introduce Elliptic Curves as the de-facto public-key algorithm for IoT and present some performance benchmarks when they are used on constrained devices. We conclude the chapter in Section 4.4.

4.1 Symmetric-key Cryptography and Advanced Encryption Standard

With symmetric-key algorithms, both communicating parties use the same secret key for forward and inverse transformations – e.g. encryption and decryption, Message Integrity Code (MIC) generation and verification. An encryption scheme consists of two publicly known algorithms E and D, such that:

$$c = E(k, m),$$

and

$$m = D(k, c),$$

where k denotes a secret key, m denotes *plaintext*, i.e. message to be encrypted, and c the *ciphertext*, ineligible representation of m. Key k has a fixed length, and the longer the key, the harder it is to guess by brute force[1]. Latest recommendations [39] declare key lengths of 128 bits appropriate for securing information whose "security life" extends year 2031. Message m can be of any given length. A *block cipher* is an encryption scheme which breaks up the plaintext messages into *blocks* of fixed length and encrypts one block at a time [103].

The block cipher that transitioned cryptography from the proprietary worlds of military, secret agencies and government applications to the wide public is Data Encryption Standard (DES). The adoption of DES in 1976 as a federal standard in the United States was unprecedented – never before had an algorithm evaluated and declared "secure" by National Security Agency (NSA) been made public [137]. DES was purposely designed to be fast in hardware, and slow in software and therefore prioritized the users who had means for costly hardware design. During the standardization process, the 54-bit key length of DES received much criticism from the community as too weak [137] which suggested that the government consciously decided to weaken the security just enough so that NSA could break it [73]. Ever since, from Edward Snowden's revelations we have witnessed that secret agencies have found more subtle ways of breaching computer security, without the explicit need to weaken the cryptographic algorithms[2]. Indeed, Bruce Schneier states [138] that although "cryptography is strong, computer security is weak".

4.1.1 Advanced Encryption Standard

The call for a DES successor was issued in 1997, and came to be known as the AES selection process. In October 2000, Rijndael cipher was selected and became known as AES. The cryptographic community has praised the selection process and the winner, AES, is as-of-today deemed appropriate for protection of US top-secret documents, in its 192-bit or 256-bit key length variants.

AES is a block cipher which operates on 16-byte block size, and uses three possible variants for key size: 128, 192 or 256 bits. In IoT application and this thesis, we consider the 128-bit key length as sufficient, as it provides for "security life" far greater than the expected lifetime of protected information. AES uses a series of permutations and substitutions, and therefore executes fast when implemented in both hardware and software. We only attempt here to give a high level description of the algorithm and for details the reader should refer to the book of Daemen and Rijmen [27].

[1] Brute force attack consists in exhaustive search over all the possible values of the secret key.
[2] Although such attempts are also not uncommon, for example with Dual EC random generator.

Inner Workings of AES

AES consists of the repeated application of the round transformation on intermediate result, called *state*. Number of rounds N_R depends on the key length L_K:

$$N_R = \begin{cases} 10, L_K = 128\ bits \\ 12, L_K = 192\ bits \\ 14, L_K = 256\ bits \end{cases}$$

Block size L_B is constant and is equal to 128 bits (16 bytes). Algorithm 1 depicts the operations of an AES encryption. The prerequisite is the `KeyExpansion` procedure that takes as input the secret key provided by the user and expands it into $N_R + 1$ keys of length L_B that are used for each round of the algorithm.

Algorithm 1 AES block encryption.

```
1: procedure ENCRYPTION(State, Key)
2:     KeyExpansion(Key, ExpandedKey)
3:     AddRoundKey(State, ExpandedKey[0])
4:     for 0 < i < N_R do
5:         Round(State, ExpandedKey[i]);
6:     FinalRound(State, ExpandedKey[N_R])
```

The actual encryption starts with the `AddRoundKey` transformation where the original 16-byte plaintext is xored with the first round key. `AddRoundKey` is followed by $N_R - 1$ iterations of `Round` transformation and one application of `FinalRound`.

Algorithm 2 `Round` transformation of AES.

```
1: procedure ROUND(State, ExpandedKey[i])
2:     SubBytes(State)
3:     ShiftRows(State)
4:     MixColumns(State)
5:     AddRoundKey(State, ExpandedKey[i])
```

Algorithm 3 `FinalRound` transformation of AES.

```
1: procedure FINALROUND(State, ExpandedKey[N_R])
2:     SubBytes(State)
3:     ShiftRows(State)
4:     AddRoundKey(State, ExpandedKey[N_R])
```

Algorithms 2 and 3 show the inner *steps* of `Round` and `FinalRound` transformations. Note that `FinalRound` differs from `Round` only in the absence of `MixColumns` step.

`SubBytes` step is the substitution step of the cipher where each byte in the state is substituted using a Rijndael substitution box (S-box). Practically, this results in

a table lookup operation for each byte of the 16 bytes of state. The main property of `SubBytes` step is non-linearity, and it is in fact the only non-linear transformation of the cipher. S-box values are carefully constructed and are paramount for the security of the cipher [27].

`ShiftRows` step cyclically shifts rows of the state represented as a 4×4 matrix over different offsets. The four offsets have to be different to achieve resistance against differential and linear cryptoanalysis. First row of the state matrix is not shifted, while second, third and fourth rows are shifted with offsets one, two and three, respectively.

`MixColumns` step is another linear transformation where each column of the state matrix is multiplied modulo $x^4 + 1$ with a fixed polynomial. The coefficients of the polynomial are selected in a way that the multiplication can be implemented very efficiently even on 8-bit processors.

`AddRoundKey` step is a simple Exclusive OR (XOR) operation of the state with a given round key. This step provides the binding of the state with the secret key. `AddRoundKey` is its own inverse.

AES decryption can be performed by using the inverses of the above steps in the reverse order. From the implementation point of view, this introduces additional code-size complexity that can be avoided by techniques that we discuss in the following sections. The inverse of `SubBytes` step adds particular overhead as another table (inverse S-box) needs to be stored in memory.

4.1.2 Block Cipher Modes of Operation

A block cipher, such as AES, encrypts plaintext of fixed size L_B, that in the AES case corresponds to 128 bits. As messages are often much longer than a single block, a need arises to use the block cipher in a certain Mode of Operation (MOP). For simplicity, let the length of message M be a multiple of block size L_B, such that it can be partitioned into t plaintext blocks, each L_B bits long: $x_1, x_2, \ldots, x_t$.

Electronic Codebook. The straightforward way of encrypting message M is to encrypt each block separately which is known as Electronic Codebook (ECB) mode of operation. To encrypt, one needs to run the block cipher t times to produce the t blocks of ciphertext.

$$c_i = E(k, x_i), 1 \leq i \leq t, \tag{4.1}$$

and to decrypt:

$$x_i = D(k, c_i), 1 \leq i \leq t \tag{4.2}$$

ECB mode has an undesirable property that identical plaintext blocks result in identical ciphertext [103]. As fragments of messages tend to repeat (network protocols headers, same application data) it is fairly easy to mount an attack on such a scheme by statistical analysis [137].

Cipher Block Chaining. A more secure encryption mode for a block cipher is called Cipher Block Chaining (CBC). In order to remove the property of ECB

that identical plaintext blocks result in identical ciphertext, CBC introduces dependency between subsequent ciphertext blocks: the i^{th} ciphertext block is obtained by encrypting the i^{th} plaintext block with $(i-1)^{th}$ *ciphertext* block. This makes sure that two identical blocks within a message result in different ciphertext blocks. However, two identical messages will still result in the identical ciphertext which can leak information[3]. To overcome this, it is necessary to introduce some *randomness* in the encryption process. For that reason, the first plaintext block of a message is xored with Initialization Vector (IV). Different IV will result in the same plaintext producing different ciphertext when encrypted under the same key k.

Algorithm 4 Cipher Block Chaining (CBC) encryption and decryption.

1: **procedure** EncryptionCBC($k,\ IV,\ x_1,\ x_2,\ \ldots,\ x_t$)
2: $c_0 = \text{IV}$
3: **for** $1 \leq i \leq t$ **do**
4: $c_i = E(k,\ c_{i-1} \oplus x_j)$
5: **procedure** DecryptionCBC($k,\ IV,\ c_1,\ c_2,\ \ldots,\ c_t$)
6: $c_0 = \text{IV}$
7: **for** $1 \leq i \leq t$ **do**
8: $x_i = c_{i-1} \oplus D(k,\ c_i)$

We depict the steps of CBC encryption and decryption in Algorithm 4. There are several interesting remarks about this mode. If an error occurs on one of the ciphertext blocks, all the subsequent blocks are useless and they cannot be decrypted. The use of IV adds communication overhead, as the party that is performing decryption needs to be aware of it. IV does not need to be secure, but it must be integrity protected, as by altering the IV, an attacker can make predictable changes in the first plaintext block that is recovered [103].

Cipher Block Chaining Message Authentication Code (CBC-MAC). Note that both CBC and ECB modes only ensure confidentiality of data. They do not provide any guarantees that data has not been modified, before the decryption takes place. Indeed, if a message is encrypted without any integrity protection, an attacker can predictably affect the plaintext by changing bits in the ciphertext. To prevent this, typically one uses keyed hash functions. The resulting hash depends on both the message and the secret key. In such a way, only parties in possession of the secret key can produce or verify the Message Authentication Code. However, this can also be achieved with a block cipher and CBC mode of encryption. Note how the last ciphertext block c_t in the CBC encryption procedure of Algorithm 4 depends on all the previous blocks of the ciphertext and *plaintext*. If a single bit is different on any bit in the message, c_t will be affected. Consider now that message is communicated in clear but c_t is sent alongside the message. The receiving party

[3]Consider for instance a military agency that recorded a ciphertext of message "attack", and learned from previous experience that it gives the aerial unit of the enemy command to launch the attack. Next time the same ciphertext appears on the air, there is no need to break the complicated encryption scheme in order to figure out what is coming.

simply needs to verify if the received c_t matches the one that is calculated locally and verify the message against alterations on the communication path. To generate a CBC-MAC, it suffices to set the IV to an all-zero vector. We depict the steps in Algorithm 5. Note that for both generation and verification of CBC-MAC, only block cipher encryption is used.

Algorithm 5 Generation and verification of CBC-MAC.

procedure GENERATEMAC($k,\ x_1,\ x_2,\ \ldots,\ x_t$)
  $c_0 = \{0\}^{L_B}$
  for $1 \leq i \leq t$ **do**
    $c_i = E(k,\ c_{i-1} \oplus x_i)$
  $mac = c_t$
  return mac
procedure VERIFYMAC($k,\ mac,\ x_1,\ x_2,\ \ldots,\ x_t$)
  $c_0 = \{0\}^{L_B}$
  for $1 \leq i \leq t$ **do**
    $c_i = E(k,\ c_{i-1} \oplus x_j)$
  if $c_t \equiv mac$ **then**
    return True
  else
    return False

Counter mode. Finally, we examine another mode of operation called Counter (CTR) that allows encryption and decryption to be performed in parallel. CTR mode uses the IV as a monotonic counter to encrypt/decrypt successive blocks. As a matter of fact, CTR mode does not pass the plaintext blocks through the block cipher encryption primitive at all. Instead, it simply xors the plaintext block with the encryption of the monotonic counter with secret key k:

$$c_i = x_i \oplus E(k,\ IV_i),\ 1 \leq i \leq t. \tag{4.3}$$

Seemingly simple, this is a powerful construct that owes its security to 1) the randomness properties of the underlying block cipher; 2) the XOR property that generates uniformly-distributed output when only one of the operators (the encrypted counter) is uniformly distributed. In fact, the "perfect security" of One Time Pad (OTP) is due to the latter property of XOR and the random one-time keys. If we extend Eq. 4.3 by xoring both sides with $E(k,\ IV_i)$, we obtain:

$$c_i \oplus E(k,\ IV_i) = x_i \oplus E(k,\ IV_i) \oplus E(k,\ IV_i), \tag{4.4}$$

and since $x \oplus\ x = 0$ and $y\ \oplus\ 0 = y$, we get:

$$x_i = c_i \oplus E(k,\ IV_i),\ 1 \leq i \leq t, \tag{4.5}$$

which is the decryption relation of the CTR mode. Note that the block cipher is only used in encryption mode.

Similarly as in OTP case, the security of CTR relies on the assumption that the same message is never encrypted with the same IV and key k twice. Consider what happens if two plaintext blocks, x and y are accidentally encrypted breaching this assumption:

$$c_1 = x \oplus E(k,\ IV), c_2 = y \oplus E(k,\ IV). \tag{4.6}$$

Since the attacker is in possession of both c_1 and c_2, he only needs to perform an XOR of the two ciphertexts:

$$c_1 \oplus c_2 = x \oplus y, \tag{4.7}$$

to get the XOR of the two plaintext blocks, which completely breaks the security, as it is now trivial to obtain x and y with statistical analysis.

To avoid this, IV is typically constructed from two parts: 1) a *nonce* that should never be reused; 2) a simple monotonic counter concatenated to the nonce, that increments with each encrypted block of a message.

4.1.3 Authenticated Encryption with Associated Data and CCM

The modes we discussed above provide either confidentiality (ECB, CBC, CTR) or data origin authentication[4] (CBC-MAC). In networking applications, confidentiality without integrity is not of much use since the attacker can introduce predictable changes in the plaintext by modifying bits of the ciphertext. Additionally, we often need a part of the message to be in clear in order to facilitate the communication (for example, addresses) while the rest of the message (payload) needs to be encrypted. The part that is in clear, however, needs to be authenticated and integrity-protected.

This is achieved with Authenticated Encryption with Associated Data (AEAD) primitives, which combine encryption with data origin authentication in a single construct. The two most popular AEAD schemes are Counter Mode Encryption and Cipher Block Chaining Message Authentication Code (CCM) and Galois/Counter Mode (GCM). We focus on CCM, due to its wide availability in IoT hardware and standards.

4.1.3.1 Description of CCM Mode

As its name states, CCM uses CTR mode for encryption and CBC-MAC for data origin authentication. CCM [37] is based on MAC-then-Encrypt paradigm, where the authentication tag (MAC) is first produced on the plaintext, and then the plaintext and authentication tag are encrypted altogether. Note that a part of the message, i.e. Associated Data, can be only authenticated. CCM requires two block cipher encryptions for each block that is both encrypted and authenticated, and one block cipher encryption per block that is only authenticated. Extension of Counter Mode Encryption and Cipher Block Chaining Message Authentication Code (CCM*) has also been defined in order to generalize CCM to the encryption-only case and avoid some vulnerabilities for variable-length authentication tags that apply to the original CCM mode.

[4] Data origin authentication is a stronger property than integrity and therefore implies it.

CCM and CCM* specifications define how to construct the vectors that serve as plaintext or ciphertext inputs to CBC-MAC and CTR modes. Since message length is often not a multiple of block size, construction of CCM vectors implies data padding. Also, CCM vectors encode the information on message length, and authentication tags in order to avoid related attacks.

In summary, CCM and CCM* provide several interesting features for constrained devices. Since they are based on CTR and CBC-MAC primitives, they only require the block cipher to be used in *encryption* mode. That means that AES implementations for constrained devices do not need to account for inverse procedures of AES steps discussed in Section 4.1.1, necessary for AES block decryption. Furthermore, CCM and CCM* do not add any overhead for padding of the message, because of the way plaintext and ciphertext vectors are constructed.

4.2 CRYPTO ENGINE: An Application Programming Interface for Hardware-Accelerated Symmetric Cryptography

In this section, we present an Application Programming Interface (API) whose main goal is to leverage the available acceleration of cryptographic primitives available in hardware and by doing so to reduce development time while maximizing the efficiency of a CCM implementation on a generic board[5][6]. CCM is the most widely used symmetric cipher on constrained devices but we also aim to support other modes that may be used by the application developer. It is important to understand that efficiency of a given CCM implementation depends on many factors relevant to chip-level and board-level design, as well as the actual application of CCM.

For instance, hardware acceleration block can be within the Microcontroller Unit (MCU), radio chip or present as a separate chip. In the latter two cases, efficiency of a CCM implementation can largely depend on chip-to-chip communication protocol overhead (e.g. Serial Peripheral Interface (SPI)). Hardware acceleration blocks embedded within the radio chip typically target usage for direct radio communication (i.e. link-layer security) and are compliant to the same standard as the radio transceiver. These, however, can still be leveraged for a generic usage, if the MCU does not provide its own crypto acceleration block.

Capabilities of different crypto hardware acceleration blocks vary (see Table 4.1). Common point is that stand-alone usage of AES is generally supported. An optimal implementation of CCM for a given board will leverage the maximum number of operations available in hardware, and complement the rest in software. While this may seem simple, in the context of an open-source project [177], it is important to

[5] CRYPTO ENGINE with implementations for CC2538 SoC and CC2420 chips makes part of the official OpenWSN project. Implementation for STM32L151 is part of the GREENNET project.

[6] CRYPTO ENGINE is based on the preliminary work of Marcelo Barros de Almeida. Author of the manuscript contributed its redesign, integration with OpenWSN project and all hardware-accelerated implementations.

Table 4.1: Hardware acceleration capabilities for AES and different modes of operation on IoT chips.

Part Number	stand-alone AES	CBC	CTR	CCM
STM32L1 [148]	yes	yes	yes	no
Testchip RF200 [c_2]	yes	no	no	no
AT86RF231 [7]	yes	yes	no	no
CC2538 SoC [157]	yes	yes	yes	yes
CC2520 [156]	yes	yes	yes	yes
CC2420 [19]	yes	yes	yes	yes

stay generic and maximize code re-use across different hardware. In the same time, such an approach renders minimal development time, as the developer can re-use the available libraries to complement the hardware-specific implementation.

Interface

Because hardware capabilities are typically delimited by different Modes of Operation, we choose to expose CCM building blocks: stand-alone AES, CBC, CTR, as well as the "highest level" call to CCM forward and inverse transforms (see Table 4.2). Different standards and applications use CCM in different context (i.e. radio frames, application data) so it is important to account for variable nonce length. For example, IEEE 802.15.4 standard uses 13-byte long nonces [62], while CCM-based cipher suites of Datagram Transport Layer Security (DTLS) use 12 bytes [100].

Table 4.2: CRYPTO ENGINE interface. API of CCM accounts for variable nonce length; CBC, CTR and AES calls are generic.

CCM
CBC encryption
CTR encryption
Stand-Alone AES

Modularity

Following Table 4.2, we organize in modules different routines of the CRYPTO ENGINE API. Modules are bound together in a static pointer structure which allows linking of different hardware/software implementations. For example, one can implement stand-alone AES acceleration for a given board and link that with software implementations of CCM, CBC, and CTR modes. Similarly, any hardware/software

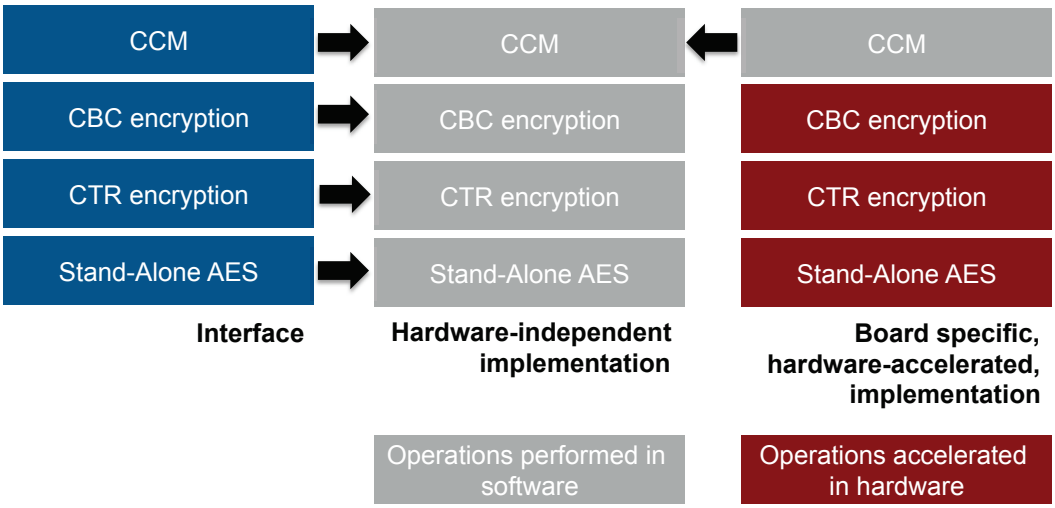


Figure 4.1: CRYPTO ENGINE implementation example with everything but CCM-specific operations accelerated in hardware.

combination is possible, depending on desired, i.e. available, level of hardware acceleration.

All-Software Implementation

Where acceleration is not possible, one needs to complement the CRYPTO ENGINE implementation for a given board with software. Hardware-independent implementation of CRYPTO ENGINE is provided as part of the OpenWSN project in order to reduce the development time.

Example Use Case

We show in Fig. 4.1 an example implementation of CRYPTO ENGINE for STM32L1 MCU. In this case, hardware provides acceleration for CBC and CTR modes. Broadly speaking, call for a CCM forward transform will result in:

1. Creation of padded CCM vectors with encoded lengths of message, MIC and nonce.
2. Call to hardware-accelerated CBC encryption routine.
3. Fetching of the last block of CBC encryption from hardware (CBC-MAC).
4. Call to hardware-accelerated CTR encryption routine.

In case of the inverse transform, steps 2 - 4 are executed in the reversed order: 4, 2, 3. In addition, one memory comparison operation is performed to verify the MIC. Note that the call to stand-alone AES is not necessary since it is automatically performed in hardware, as part of CBC and CTR routines.

4.2.1 Performance on Constrained Devices

We implemented hardware-accelerated CRYPTO ENGINE for three IoT boards: GREENNET, OpenMote-CC2538[7], and TelosB[8]. We summarize the three boards and their salient characteristics in Table 4.3. Our goal in this section is to draw conclusions on CCM performance depending on different design choices that can be useful to chip-level, board-level, and network-level designers.

We perform all measurements using a logic analyzer connected to physical pins on the board. In software, we set *high* the voltage of a MCU pin just before the execution of the desired section and *low* once the device terminates execution. Logic analyzer is connected to the same physical pin and samples its state with 8 Msamples/s rate. We record measurements by monitoring the output of the logic analyzer using the provided software.

Table 4.3: Boards used for performance evaluation of hardware-accelerated symmetric cryptography. SPI access denotes whether crypto acceleration block is accessed over SPI (Y) or it is embedded in the MCU (N).

Board	System Clock [MHz]	Hardware Acceleration Part	SPI access [Y/N]
GREENNET	32	STM32L1	N
OpenMote-CC2538	32	CC2538	N
TelosB	8	CC2420	Y

Computation Overhead of Stand-Alone AES

We first measure the computation overhead of block (i.e. 16 bytes) AES encryption (see Table 4.4). For software AES, we use the open-source implementation by Uli Kretzschmar that is optimized for MSP430 processors but can run on a generic MCU. As a consequence, AES software results are not optimal for GREENNET and OpenMote-CC2538 devices, but are rather presented for comparison purposes with hardware-accelerated code. We leverage compiler optimization and use the $-Os$ flag in the corresponding toolchains.

From Table 4.4, we can notice that the gain factor obtained by hardware-acceleration for GREENNET and OpenMote-CC2538 is around 14. The cost of accessing the crypto acceleration block is high – it makes up for at least 55% of the overall latency. STM32L1 (GREENNET) executes the same software AES code 23%

[7]OpenMote-CC2538 [112], released in 2013, is the flagship board of OpenWSN project with CC2538 System on Chip (SoC) at its core. Radio transceiver and MCU are integrated on the same chip and share RAM memory.

[8]TelosB [155], released in 2004, is the second generation of historic Berkeley motes, based on MSP430 MCU and CC2420 radio, connected over SPI.

Table 4.4: Computation overhead of block AES encryption.

Board	Software	Hardware-accelerated		
	total [µs]	key expansion and encryption	I/O access	total [µs]
GreenNet	212	44.9%	55.1%	16
OpenMote-CC2538	275	13.9%	86.1%	19
TelosB	2030	5.2%	94.8%	1114

faster than CC2538. In terms of overall latency, STM32L1 performs one hardware-accelerated AES encryption 16% faster than CC2538, but the time distributions reveal an interesting property. Access to the crypto block is more prohibitive on CC2538, but its AES implementation seems more efficient than the one of STM32L1. CC2538 spends around 1.9µs on loading and expanding the AES key and around 0.8µs for one AES encryption.

TelosB executes a block AES encryption in software in 2ms. Compared to more recent boards and MCUs, we can see how computation capabilities have evolved over the years: factor of 4 degradation is to be expected due to the lower system clock frequency (i.e. 8MHz vs 32MHz), while the rest is due to differences in MCU architectures. When hardware acceleration of CC2420 is used, one obtains an improvement factor of 2. However, most of the delay (95%) comes from SPI access.[9] When software execution time is compared to pure AES encryption/key expansion in hardware, we can notice a 35 improvement factor.

Computation overhead of CCM

We further quantify the computation overhead of CCM transform using the same boards. As a use case for AEAD cipher, we consider the largest message that can fit IEEE 802.15.4 radio frame with security enabled: 121-bytes[10]. Out of the 121 bytes, 30 bytes are only authenticated, while 91 bytes are both authenticated and encrypted. We present these results in Table 4.5.

Faster execution of software AES block encryption on STM32L1 propagates to the CCM use case: GreenNet performs 23.9% faster than OpenMote-CC2538.

With hardware acceleration, we can observe an interesting result. Even though single block encryption on STM32L1 is 16% faster, OpenMote-CC2538 performs better in the CCM use case. To understand why, we have quantified different components contributing to the overall latency in Table 4.6. This is due to several factors:

1. CC2538 provides full CCM support in hardware, including creation of padded

[9]For TelosB, we use the default SPI configuration of OpenWSN project with ∼500 Kb/s rate.

[10]Maximum Transmission Unit (MTU) of IEEE 802.15.4 is 127 bytes, but one needs to account for 2 bytes of Cyclic Redundancy Check (CRC) and minimal MIC length of 4 bytes.

Table 4.5: Computation overhead of CCM forward transform on 121-byte message, where 91 bytes are encrypted/authenticated and 30 bytes are only authenticated. This use case results in 9-block CBC encryption to produce the MIC, and 7-block CTR encryption for confidentiality.

Board	software [ms]	hardware [ms]
GREENNET	3.476	0.257
OpenMote-CC2538	4.567	0.056
TelosB	34.095	5.325

CCM vectors that with STM32L1 we had to perform in software: this results in 19% software processing delay.

2. Absence of CCM support on STM32L1[11] necessitates to perform I/O access for CCM vectors that are 16-blocks long, instead of transferring the 8 blocks of the original message (121 bytes). In addition, we had to read the intermediate results of CBC encryption, when we only needed the last block (CBC-MAC). This results in 40% I/O access delay. Note that the figure of 40% also accounts for software processing, that is a side-effect of such design.

3. Multiplicative effect of AES block encryption performance with STM32L1.

Table 4.6: Delay components contributing to total CCM latency with hardware acceleration.

Board	hardware encryption	I/O access to crypto block	software processing
GREENNET	40.87%	40.04%	19.09%
OpenMote-CC2538	34.2%	45.6%	20.3%
TelosB	5.05%	87.95%	7.0%

TelosB software results (see Table 4.5) can be extrapolated from the execution time of block AES encryption. Results in hardware, however, confirm the advantage of having full hardware-accelerated MOP implementation: encryption of 16 blocks only results in 5x degradation in respect to a single AES encryption. I/O access to the crypto block proves to be the most prohibitive operation – it accounts for 88% of the overall latency. However, as hardware acceleration is part of the CC2420

[11] Another hardware part from STMicroelectronics is available that trades off space for better computational performance, provides full CCM implementation in hardware, and performs an AES block encryption an order of magnitude faster than its smaller counterpart used on GREENNET nodes.

radio chip, its main envisioned application is link-layer security. In that context, when CC2420 is used to transmit a frame, it can perform CCM forward transforms on the frame *already* loaded in the radio buffer and thus almost completely avoid the crypto-specific I/O overhead[12]. On the reception side, however, it is necessary to read over SPI parts of the radio frame that are sent in clear and decide which key/nonce to use for decryption of the frame still loaded in the radio buffer. Nevertheless, such design enables TelosB to secure radio frames even in tightly synchronized network protocols, such as Time-Slotted Channel Hopping (TSCH).

To put the above figures in energy-constrained context, in Table 4.7 we estimate the energy equivalent number of bytes transmitted over the radio during the example CCM use case. For energy estimation, we use datasheet consumption values of different components that are summarized in Tables 2.1 and 2.2 and assume that MCU is in sleep mode while radio is transmitting. An important point to note when comparing GreenNet results with OpenMote-CC2538 is extremely low consumption of prototype Testchip RF200 radio which consumes less in transmit mode than commercially available STM32L1 MCU at 32MHz.

Table 4.7: Energy equivalent number of *bytes* transmitted over IEEE 802.15.4 radio during software/hardware CCM forward transform on 121-byte message.

Board	**software**	**hardware without I/O**	**hardware**
GreenNet	131.24B	5.82B	9.70B
OpenMote-CC2538	83.35B	0.56B	1.02B
TelosB	786.81B	14.81B	122.88B

A common point for the three platforms is that software processing of CCM is inefficient. When performed in hardware, for every 121 bytes encrypted/authenticated with CCM, energy-wise one can transmit from 1.02 to 14.81 bytes over the radio (we consider link-layer encryption use case with TelosB).

Related Work

Some of the related work tackles optimized hardware designs of AES and CCM for constrained devices [51, 57, 147]. Hamalainen *et al.* [51] describe a compact and energy-efficient hardware implementation of IEEE 802.15.4 security, and show its advantage in terms of execution speed and energy consumption. Huai *et al.* [57] and similarly Song *et al.* [147] design an energy-efficient CCM hardware architecture for IEEE 802.15.4 networks. An interesting work of Otero *et al.* [114] compares different hardware implementations approaches for the AES block cipher and their performance in terms of throughput, power and memory overhead. In their performance evaluation, they hint that throughput of a hardware AES block

[12] One still has to transfer 13-byte nonce for each frame, and select/write the corresponding key.

on evaluated MSP430 SoC drops by 44% due to I/O operations, which is consistent with our findings.

Raza *et al.* [126] evaluate different symmetric primitives implemented in software and hardware on TelosB as part of their study on higher level protocols and conclude that energy consumption can be reduced 50% by leveraging hardware acceleration. They derived this figure from the overall performance of the upper-layer protocol but our results suggest that pure gains in terms of cryptographic operations are much higher. The focus and contribution of our performance evaluation was to dissect different delays contributing to the overall latency in order to provide useful inputs to chip-level and board-level designers. We study in more details the effect of cryptographic algorithms on upper-layer protocols in Part II.

In the context of other open-source projects, Contiki recently released support for security with its 3.0 release. Contiki uses two modules, one for AES and the other for CCM which also allows linking of hardware-accelerated implementations for different platforms. However, this approach does not allow clean code re-use in cases when hardware partially supports various MOP, like CTR or CBC.

Conclusion

Our performance study revealed that hardware acceleration of symmetric-key cryptography is a must, both in terms of energy and latency. Crypto acceleration blocks within the MCU are more performant due to the lower I/O access latency. Flexibility is also very important as both AES and different MOP such as CCM can be used in various contexts, ranging from link-layer to application-layer security. We could learn from TelosB example that SPI access can incur significant overhead that may cancel out the benefits of provided hardware acceleration. In such cases, hardware acceleration makes sense only if complete, use case specific MOP implementation is provided such that MCU does not need to handle each block. Additionally, benefits of a full MOP implementation vs. implementation of different building blocks were evident in the STM32L1 case, where the use of hardware accelerated CTR and CBC modes resulted in unnecessary I/O transfers and associated processing that significantly degraded the overall performance.

4.3 Public-Key Cryptography and Elliptic Curves

Symmetric-key primitives are very efficient but require shared knowledge of a secret key. Everyone in possession of this key has unconditional access to protected data, either to decrypt and verify the authentication *or* to encrypt new data and generate a valid authentication tag. This poses challenges in terms of 1) key exchange, i.e. how to establish the secret key; 2) group communication, because all members of the group need to be fully trusted.

Public-key techniques [31] revolutionized the field of cryptography and enabled secure key exchange between parties that are communicating for the very first time. The basic notion of public-key cryptography is that keys can come in *pairs*, an

encryption key and a decryption key, and that it could be infeasible to generate one key from the other [137]. The challenge of public-key constructs is the definition of "infeasible" that is both practically useful and secure.

4.3.1 Integer Factorization and RSA

The most-widely known public-key algorithm is Rivest, Shamir, Adelman (RSA) [86], named after its inventors Ron Rivest, Adi Shamir and Leonard Adleman. RSA dues its security to the difficulty of factoring large numbers. Key generation consists of choosing two random large prime numbers p and q and computing their product $n = pq$. The encryption key e is selected such that e and $(p-1)(q-1)$ are relatively prime, and the decryption key d is computed such that:

$$ed = 1 \ mod \ ((p-1)(q-1)). \tag{4.8}$$

Integers e and n represent the public key and d is the private key. Block i of message m is encrypted by exponentiation with e modulo n:

$$c_i = m_i^e \ mod \ n. \tag{4.9}$$

Decryption consists of exponentiating the ciphertext block c_i with decryption key d:

$$c_i^d \ mod \ n = (m_i^e)^d = m_i^{ed} = m_i \ mod \ n. \tag{4.10}$$

Block size is chosen such that $m_i < n$, which allows to recover the original plaintext. Note that a message encrypted with e can only be decrypted with d. Since d is private, anyone can encrypt a message but only the party in possession of d can decrypt it. Security of RSA relies on intractability of the RSA problem: given n and e, find an integer m_i such that $m_i^e \equiv c_i \ mod \ n$ [103]. For the security level of approximately 128 bits, RSA requires the usage of modulus n of 3072 bits [39], while 2048-bit modulus commonly used today, provides security level of approximately 112 bits.

RSA is not well suited to constrained devices due to 1) large keys that may need to be exchanged over the network; 2) computational complexity that is prohibitively expensive on constrained devices (100-10000× of the symmetric counterpart) [137, 173].

4.3.2 Discrete Logarithms in a Finite Field and Elliptic Curves

An operation frequently used in cryptography is modular exponentiation:

$$y = g^x \ mod \ p, \tag{4.11}$$

and it can be efficiently computed. Determining x, given y and domain parameters g and p is known as the Discrete Logarithm Problem (DLP) [52]. Diffie-Hellman key agreement, ElGamal encryption and signature schemes, Digital Signature Algorithm

(DSA) are some of the examples of public-key schemes that rely on intractability of DLP.

Schemes based on DLP can be implemented using elliptic curves, in which case the problem is known as Elliptic Curve Discrete Logarithm Problem (ECDLP). The best algorithms known to solve the ECDLP have fully exponential running time, in contrast to the subexponential-time algorithms known for the integer factorization problem [52]. The consequence is that schemes based on ECDLP can attain the same level of security with significantly smaller key sizes than their integer factorization counterparts, such as RSA. For instance, 128-bit security level with Elliptic Curve Cryptography (ECC) requires approximately 256-bit keys, which is a factor of 12 improvement over RSA. Smaller keys result in better efficiency, both in terms of computational complexity and communication overhead.

Elliptic curves can be defined over a generic field $\mathbb{F}$, such as the familiar number systems of rational, real or complex numbers together with the operations of addition and multiplication. In cryptographic applications, elliptic curves are defined over *finite fields*, also known as Galois fields. For instance, let p be a prime number. The set $\{0,\ 1,\ \ldots,\ p-1\}$ with addition and multiplication performed modulo p is a finite field of order p, denoted $\mathbb{F}_p$. An elliptic curve E over $\mathbb{F}_p$ is defined by an equation of the form:

$$y^2 = x^3 + ax + b, \tag{4.12}$$

where $a, b \in \mathbb{F}_p$ and the discriminant is not equal to zero. A pair (x, y), where $x, y \in \mathbb{F}_p$ is the point on the curve, if (x, y) satisfy Eq. 4.12 [52].

Parameters prime p, equation E, a point on the curve P and its order[13] n characterize a curve for cryptographic use and represent public information. A private key is generated by selecting an integer d randomly from $[1, n-1]$. The corresponding public key Q is simply $Q = dP$. Finding d from Q and public parameters of the curve is in fact the ECDLP on whose intractability security of different constructs depends. The length of parameter n therefore influences the desired security level, which is roughly half the length of n.

Elliptic curves can be used to implement various public-key algorithms that provide services such as key exchange, confidentiality, or data origin authentication. Such mechanisms depend on arithmetic operations of points on a specific curve that essentially depends on the underlying finite field. The two basic operations are *point multiplication* and *point addition*. Point addition is a relatively inexpensive operation, for example a hardware-accelerated implementation of a 256-bit curve can consume roughly 9×10^4 clock cycles. Point multiplication, however, is much more prohibitive and in the 256-bit scenario can consume roughly 5.5×10^6 clock cycles.

Some fields were found to be weak [102] in terms of the ECDLP and security of *any* elliptic curve over the affected fields is significantly reduced.

[13] Smallest positive integer n such that $g^n = 1$ is called the order of g [52].

4.3.2.1 Elliptic Curve Digital Signature Algorithm

In Section 4.1.2, we examined a symmetric construct that provides data origin authentication – CBC-MAC. Problem with symmetric data authentication schemes is that sender cannot be differentiated from receiver(s). This is elegantly solved with asymmetric techniques and digital signatures where only the party in possession of the private key can generate the signature while everyone in possession of the public key can verify it. We consider here Elliptic Curve Digital Signature Algorithm (ECDSA) that is very fit for constrained devices due to low computational complexity in comparison to other public-key schemes. Signature length of ECDSA is twice the length of n, prime order of P, and is roughly four times the desired security level.

Algorithm 6 ECDSA signature generation and verification [52].

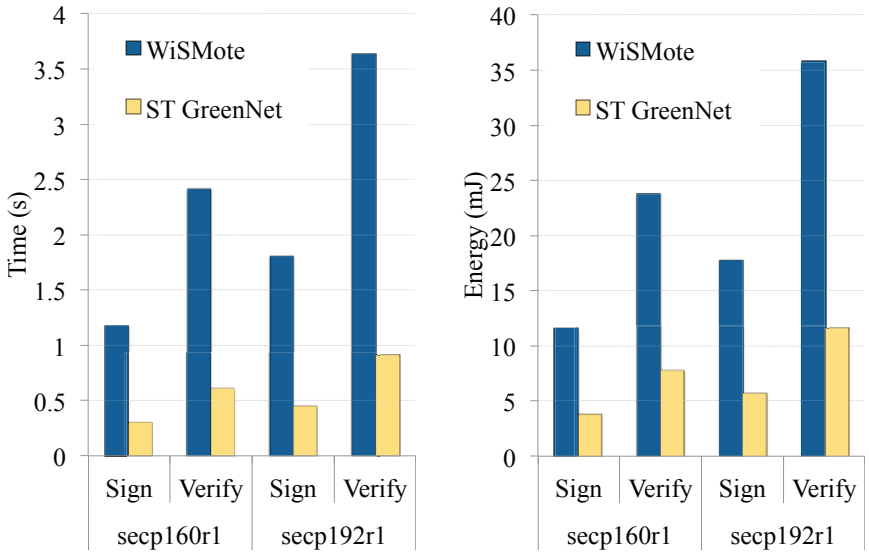
procedure SIGN(private key d, message m, public parameters p, E, P, n)
  Select random k from $[1, n-1]$
  $k \times P = (x_1, y_1)$, convert x_1 to an integer $\overline{x_1}$
  $r = \overline{x_1} \bmod n$, if $r = 0$ then go to step 2
  $e = Hash(m)$
  $s = k^{-1}(e + dr) \bmod n$. if $s = 0$ then go to step 2.
  return $(r,\ s)$
procedure VERIFY(signature $(r,\ s)$, public key Q, message m, public parameters p, E, P, n)
  Verify that r and s are integers in $[1, n-1]$. If not true, **return** `false`.
  $e = Hash(m)$
  $w = s^{-1} \bmod n$
  $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$
  $X = u_1 \times P + u_2 \times Q$
  If $X = \infty$ then **return** `false`
  Convert x_1 of X to an integer $\overline{x_1}$; $v = \overline{x_1} \bmod n$
  If $v = r$ then **return** `true`
  Else **return** `false`

We depict the steps of ECDSA signature generation in Algorithm 6 and denote with $\times$ expensive point multiplication operation. Note that the signing operation requires a single point multiplication, while the verification procedure performs the point multiplication twice. This is the main reason why the performance of ECDSA verification algorithm is much worse than that of the signing counterpart. For a detailed proof why the signature verification steps in Algorithm 6 are indeed correct, the reader should refer to Hankerson *et al.* [52].

Performance on Constrained Devices. Figs. 4.2(a) and 4.2(b) present computation and energy benchmarks of the ECDSA primitives (secp160r1 and secp192r1 elliptic curves) on WiSMote[14] and GREENNET boards, discussed in Chapter 2. We

[14]WiSMote board [179] is based on 16-bit MSP430 (series 5) MCU with 16 KB of RAM and an

use the TinyECC open source library [95] and there is no hardware acceleration involved. We can see that the use of a 32-bit MCU on GreenNet boards reduces the computation time by a factor of 4, which translates into a reduction in the consumed energy by a factor of 3.084 (as the 32-bit STM32L1 consumes 29.7% more than 16-bit MSP430 in active mode). The computation overhead ranges from 0.3 to 0.9 seconds for GreenNet and from 1.18 to 3.63 seconds for WiSMote.



(a) Computation time.

(b) Energy consumption at 2.8V.

Figure 4.2: ECDSA computation and energy benchmarks at 21.3 MHz for 16-bit (WiSMote) and 32-bit (ST GreenNet) hardware platforms.

We can notice how computation capabilities on commodity hardware have changed over the years and that ECC primitives are already feasible, even when implemented purely in software with no hardware-specific instructions. That said, assembly optimizations can further improve the software performance by a factor of 2 to 3. Hardware-acceleration parts are becoming increasingly available [157], typically leading to improvements by at least one order of magnitude over software.

4.4 Conclusion

We conclude that public-key primitives based on ECC are much more affordable than the common wisdom suggests, even on commodity IoT hardware. Availability of hardware acceleration parts further reduces their cost, making them an ideal candidate for IoT security protocols. Nevertheless, they should be used with care

802.15.4-compatible CC2520 radio transceiver.

and to complement cheaper symmetric primitives that are already widely supported in hardware, in which case they introduce a negligible overhead.

We explore how the discussed cryptographic primitives are used to secure the IoT communication stack in Part II.

Part II

Network Security and Energy Efficiency: Pick Two

Chapter 5

Security Challenges and State of the Art

Network security is one of the most prominent applications of cryptography. We leverage cryptographic algorithms as fundamental blocks to build secure distributed systems, such as the Internet. It turns out that the attribute "secure" is hard to define and often has an inherent binding with the actual system that is to be secured. Still, there have been various efforts [154, 145] on classifying and precisely defining different architectural elements, high level goals, services or simply terminology in the computer security arena.

We first overview some basic terminology in Section 5.1. Then, we discuss the traditional types of attacks on network entities and put these in the context of Wireless Sensor Networks (WSNs) and Internet of Things (IoT) in Section 5.2. In Section 5.3, we summarize the research and standardization efforts around secure WSNs and IoT. We conclude the chapter in Section 5.4.

5.1 Terminology and Definitions

In the following, we introduce the basic security terms that are useful for comprehension of this thesis. We use definitions from RFC 4949 [145].

- *Security service* – A processing or communication service that is provided by a system to give a specific kind of protection to system resources.
- *Security mechanism* – A method or process that can be used in a system to implement a security service that is provided by or within the system.
- *Adversary, attacker* – An entity that attacks a system, or an entity that is a threat to the system.
- *Attack* – An intentional act by which an entity attempts to evade security services and violate the security policy of a system.
- *Access Control* – protection of system resources against unauthorized access.
- *Authenticity* – The property of being genuine and able to be verified and be trusted.
- *(Data) Confidentiality* – The property that data is not disclosed to system entities unless they have been authorized to know the data.

Additionally, we use terms *Message Integrity Code (MIC)* and *authentication tag* interchangeably to denote a cryptographically secure checksum that is typically appended at the end of a message to provide data origin authentication and integrity. It is also known as Message Authentication Code (MAC) but we avoid this usage due to a conflict with Medium Access Control (MAC).

5.2 Threat Models and Typical Attacks

Threat model formally defines capabilities of the attacker and security of a system can be studied within a given model. Internet protocols typically consider the traditional Dolev-Yao model [32] where the attacker has full control over the network. More precisely, the attacker can:

- Intercept messages,
- Modify messages,
- Block messages,
- Generate and insert new messages.

It is important to understand that cryptographic algorithms are considered "perfect" and the attacker can decrypt/forge a message only if he possesses the corresponding key. In the network context, "message" corresponds to a Protocol Data Unit (PDU) of a layer under study. For instance, if we consider security solution at the link layer, message corresponds to a radio frame.

Traditionally, there are two typical classes of attacks:

- Passive attacks: Such as eavesdropping and traffic analysis, where the attacker gains knowledge on ongoing communication by passive means. For instance, if messages are sent in clear, attacker is able to read full message content. If network messages are encrypted, attacker may still be able to infer some information by studying communication patterns, timing, message length.
- Active attacks: Attacker actively participates in the communication by replaying old messages (replay attack), modifying messages and playing Man in the Middle (MITM), pretending to be another entity in order to gain unauthorized access to a resource and similar. A particular class of active attacks are Denial of Service (DoS) attacks, where the attacker's ultimate goal is to disrupt the availability of a network service, such as the alarm notification, typically by exhausting physical resources (memory, energy, bandwidth) on the target node.

An important point to note is that the Dolev-Yao model is a formal model that does not take into account physical compromise of a node. Therefore, research around WSNs [70, 169, 6, 130, 22] has often taken into account a more powerful,

Byzantine attacker [10]. In such scenarios, attacker has access to local cryptographic material on the node and we cannot rely on cryptographic techniques to prevent attacks [130]. Indeed, Byzantine attacker can compromise a set of nodes and through them inject false data that passes all cryptographic checks.

Becher *et al.* [11] conclude that physical compromise of a device in order to extract keying material and obtain full control over it, as assumed by the Byzantine model, is not as easy as often perceived in WSN literature. It requires costly equipment, expert knowledge on hardware and hard determination of the attacker. An interesting observation of this study is that such attacks often require that a mote be removed from the network for a non-trivial amount of time making detection of unusual activity via neighbor discovery protocols a simpler approach than specialized Byzantine-tolerant schemes. Common sense practices, such as disabling IEEE 1149.1 JTAG (JTAG) port or Bootstrap Loader (BSL) once the product is deployed, go a long way towards making attacks in the field more difficult.

We do recognize that in many IoT deployments, devices will be physically available to the general public and as such, system designers should take into account the threat of a physical compromise and extraction of the keying material. However, security mechanisms that protect against a Byzantine attacker are orthogonal to the mechanisms that we consider and study in this thesis.

That said, we emphasize that final IoT products should either have hardware-level or software-level protection against physical tampering, i.e. tamper-resistant packages or schemes to detect unauthorized access to the hardware [11, 96].

In the following, we survey some of the typical attacks on WSNs that have attracted attention of research community over the last decade. The more recent efforts of integrating WSNs with the Internet have enabled additional security and privacy concerns. We distinguish accordingly and brief on the main concerns in each scenario.

5.2.1 Wireless Network Threats

Physical Jamming. The most basic attempt to disrupt the network service is the attack on physical resources – the radio channel. Attacker can generate high-powered signal that will interfere at different receivers in the network and increase the error rate, possibly completely disrupting wireless operation [93, 124, 88]. This DoS attack is often called *jamming* and is mostly a concern in military scenarios. Common defense is channel hopping [63] that increases the bandwidth attacker has to jam, which can require a substantial power supply and thus make the attack less practical. Also, network-level redundancy can help in order to route around the jammed area.

Traffic Injection. Injecting false traffic in the network can have multiple consequences. Firstly, it is possible to affect network applications, e.g. by introducing a bogus temperature reading to trigger the Heating, Ventilation, and Air Conditioning (HVAC) system, or even to directly control an actuator, such as the pressure regulating valve in the industrial automation system. Similarly, one can obtain full

control over the network by forging network maintenance packets [71], e.g. beacons, and corrupting neighbor tables of the nodes. Secondly, attacker can launch easy DoS attacks by generating significant traffic loads that can cause network collapse in terms of depleted energy due to multihop forwarding or throughput.

First-level protection against such attacks is link-layer security – network nodes should not accept any radio frame other than those secured with link-level keys they possess locally. At the application level, access rights should be properly configured in order to limit the damage if one of the nodes in the network is compromised. For instance, node measuring the temperature should not be allowed to issue pressure valve regulating commands. Second-level defense is common sense programming – if some of the network nodes gets compromised and starts injecting cryptographically-valid traffic, one should locally check the rate at which it is forwarding packets or performing local operations instead of blindly following the protocol.

Attacks on Join Protocol. Link-layer security protects the wireless network in "steady" state, when all the nodes have joined and have been provisioned with necessary keying material. Before we admit a new node in the network, it is necessary to perform some checks. For example, joining a new Wi-Fi network requires the user to type in the Pre-Shared Key (PSK), which is then used to derive link-layer keys. More precisely, the joining node and the Access Point (AP) authenticate each other on the basis of this PSK. In WSNs and IoT in general, the principle is similar with the difference being that the PSK is often pre-configured in the firmware of the device by some out-of-band means. Join protocols are technology-specific but some common points exist [150, 131, 149]: 1) the joining node may initiate the join protocol multiple hops away from the gateway; 2) several messages may need to be exchanged between the joining node and the gateway before the "admittance" decision can be made. This necessitates that intermediate nodes in the mesh forward the messages that may come from a rogue joining node (attacker), which opens up the possibility of DoS attacks. Although this threat can never be fully neutralized, a common strategy is to minimize the damage a potential attacker can do. As such, one may ensure that joining messages do not instill state information in the network and can control the rate at which intermediate nodes forward join protocol messages. Caching at network edges is also a means to avoid unprotected traffic to traverse the network [149]. The latter is often conflicting with centralized Authentication, Authorization, and Accounting (AAA) systems, as it puts trust on individual network nodes that may be compromised.

Attacks on Routing Protocol. Routing aspects of WSNs have long been an interesting topic for scientific study [71, 119, 110, 1]. Due to their distributed nature, WSNs are prone to attacks that involve an attacker that can for example 1) selectively forward messages if it is within the network, or jam radio transmissions and cause collisions from the outside; 2) advertise false routes in order to attract the surrounding traffic and create a *sinkhole*, 3) present multiple false identities to other nodes in order to reduce effectiveness of fault-tolerant schemes; 4) create radio "tunnels", so called "wormholes", between two distant parts of the network in order to appear closer to the gateway and create a sinkhole at the other end of the

tunnel. Such attacks can only partly be neutralized by using link-layer security in order to reject radio frames coming from the outside. When an attacker is inside the network, i.e. a compromised node, defense requires careful design of the routing protocol that takes security into account from the beginning [71]. From the point of view of confidentiality and data origin authentication, such attacks are defended using end-to-end security mechanisms, where even the on-path attacker is not able to modify or read the data, as it is not in possession of the end-to-end keys.

Privacy issues. Sensor and actuator networks that make part of our daily life bring along various privacy issues. While management of data collected by these networks in itself represents a privacy concern, we focus on information that may leak to an outsider. Obviously, data confidentiality at the link layer (protected radio frames) is the first step to improve user's privacy. In many IoT scenarios, however, radio communication alone suffices to reveal some information about the user. For example, a presence sensor may initiate radio communication when a person enters a building [164] or a light switch may indicate that the state has been toggled by emitting a radio frame. Typical defense would involve injecting dummy traffic in the network but that may not always be feasible due to the local energy constraints.

5.2.2 Internet Threats

We noticed in the previous section that link-layer security presents the first line of defense against typical attacks launched in the radio range. The main disadvantage of link-layer security is that each node on a potentially multi hop path[1] needs to be trusted. Limits of link-layer security are typically overcome by establishing an end-to-end security channel between the node and final destination, whether the destination is the network gateway or a device outside the local network.

Indeed, as if large networks of constrained devices did not have enough security concerns on their own, IoT brings Internet Protocol (IP) connectivity to each device and makes it potentially accessible globally. However, there are techniques which can be used to ensure IP connectivity while the constrained nodes are well protected behind the gateway. Most notably, IoT gateways can 1) provide firewall and cautiously filter packets that are forwarded to the constrained part; 2) run proxies that cache latest sensor readings and serve cached readings to external clients, that perceive transparent communication with the constrained node; 3) run application-level services and completely prevent the communication of constrained nodes with the outside. To date, scenario 3) has been most widely deployed. In that case, network gateway performs all security-related operations, such as authentication, authorization and/or access control on behalf of the nodes and remote users. Attacks to obtain control of the gateway are therefore the most attractive to external attackers, but in our context are not interesting as they are defended using traditional Internet protocols (TLS, OAuth).

[1] If we consider sinkhole and wormhole types of attacks, the attacker may very well be on *every* path in the network.

5.2.3 Application Requirements

Typical security requirements of IoT applications are similar to those in the traditional Internet:

1. **End-to-end security**. From the perspective of the user[2], assurance that the information (sensor readings, actuator status, possibly low-power video surveillance) indeed comes from the network (or nodes) in question, and is not forged by the attacker. In the same time to preserve privacy, confidentiality of data in transit is required.

2. **Authorization and Access Control.** Authorized (remote) access to the abstracted physical resources in multi-user settings.

In Chapter 8, we discuss why solutions adapted for the traditional Internet are not necessarily suitable candidates for IoT and then attempt to meet these requirements by proposing a system level architecture.

5.3 State of the art

Research and standardization efforts around secure IP-based WSNs follow the TCP/IP architectural model by having security features embedded in different layers of the protocol stack. In this section, we survey the state of the art accordingly and present the most relevant literature for security-related contributions of this thesis.

5.3.1 Link-Layer Security and IEEE 802.15.4

First release of IEEE 802.15.4 standard from 2003 included support for confidentiality, data integrity, replay protection, and basic access control through its security extension based on Counter Mode Encryption and Cipher Block Chaining Message Authentication Code (CCM) mode. Soon after, Sastry *et al.* published an analysis discussing main concerns that may arise in practical implementations [136]. Most notably, they discuss insecurity of the cipher suite that only provides encryption without data integrity and possible DoS attack that can be mounted with a single forged radio frame. It is interesting to note that the cipher suite in question will finally be removed from the specification in the 2015 revision of the standard – more than a decade later. The authors also discuss problems with Initialization Vector (IV) management to avoid nonce reuse, as well as security concerns with different keying schemes. These have mostly been addressed in 2006 and 2011 revisions of the standard.

Performance of IEEE 802.15.4 link-layer security has ever since been an interesting topic for study. Many authors [28, 24, 3, 126] analyze the impact of IEEE 802.15.4 security processing on network performance. What is interesting is that

[2]User in this context simply represents the consumer of the information. It does not necessarily imply a human user – it can be another (constrained) node outside of the local network.

conclusions of these studies are somewhat contradictory and often related to the methodology and idealization of practical issues.

On one hand, authors that use analytical approach to model energy consumption [28, 109] or simulations [24] observe significant degradations to network performance. For example, Daidone *et al.* [28] report energy consumption that is 61% higher when security features of IEEE 802.15.4 are used. Similarly, Mura *et al.* [109] conclude that due to longer packets and intense calculations, security operations may represent the dominant part of the overall consumption – up to 90% depending on payload size and ciphers used. These figures, on their own, would make any system designer think twice[3] before enabling security features.

On the other hand, Altolini *et al.* [3] report on a hardware-accelerated implementation and present results that lead to 2% increase in energy consumption on AVR XMEGA AU based board. Same authors discuss that if software-based implementation of Advanced Encryption Standard (AES) is used, overall energy penalty increases up to 25%. In our studies, we find the cost of IEEE 802.15.4 security in the context of energy harvesting devices to range from 1.75% to 3.96%. In Chapter 6, we discuss in more details the origins of such discrepancies.

Raza *et al.* [126] confront the performance of IEEE 802.15.4 link-layer security with that of the end-to-end IPsec protocol. They show that IPsec outperforms link-layer security with increasing payload size and increasing number of intermediate hops. Although such result is indeed interesting, we believe that the two mechanisms serve very different requirements and that their direct performance comparison might be misleading.

Other authors tackle different aspects of key exchange schemes. Meulenaer *et al.* compare the key exchange energy cost of a Kerberos trusted third-party system against an Elliptic Curve Diffie-Hellman (ECDH)-Elliptic Curve Digital Signature Algorithm (ECDSA) public-key protocol [29]. In their evaluations, authors clearly distinguish the cost of the cryptographic processing from the communication cost and show that Kerberos outperforms ECDH-ECDSA key exchange by one order of magnitude. Khan *et al.* [78] evaluate a secure data exchange protocol built on top of the IEEE 802.15.4 layer, including a key exchange mechanism using the simulation engine Artifex. Mišić [105] analytically models different key exchange mechanisms and evaluates their impact and computation complexity on the network lifetime.

5.3.2 End-to-End Security at the Network Layer

Ever since the efforts on integrating Wireless Sensor Networks with the Internet have begun, the so-called blanket coverage at the network layer has been considered a potential solution to provide end-to-end security services [98]. The literature widely discussed the feasibility of porting the IPsec protocol suite to smart objects [49, 132, 46, 125, 126, 47]. The authors mostly evaluated the processing overhead and energy requirements of different cryptographic suites used by IPsec, but also the memory footprints and system response time [125, 126]. Even though it was

[3] More likely infinite number of times.

initially considered too heavy for constrained environments, these results led to the common conclusion that a lightweight version of IPsec is a feasible option.

In the Internet, IPsec mostly secures Virtual Private Networks (VPNs). Being at the network layer, it is perfectly suited for such applications where "blanket" coverage is actually desirable (enterprise networks for example). However, as it commonly resides in the Operating System kernel, it is impractical for typical IoT applications. The requirement that an end user needs to configure the host Operating System and IPsec for securing communication with smart objects would probably result in questionable security practices. Moreover, integrity at the network layer would prevent any protocol mappings. Namely, as the IP payload is being authenticated, there would be no way of performing Hypertext Transfer Protocol (HTTP)/Constrained Application Protocol (CoAP) mapping at the network gateway. CoAP, however, has been designed from the very beginning to facilitate this for legacy hosts in the Internet.

5.3.3 End-to-End Security at the Transport Layer

Impracticality of IPsec has been overcome in the Internet by introducing the security services just below the application layer, in the form of Transport Layer Security (TLS)/Secure Sockets Layer (SSL). The wide and successful use of this model in the Web has also suggested its use in IoT. Indeed, the first proposal on using SSL for smart objects, nicknamed Sizzle, came in 2005 from SUN Microsystems [50]. The authors evaluated the HTTPS stack that leverages assembly optimized implementation of Elliptic Curve Cryptography (ECC) as a public-key algorithm. At the time of the publication, however, there was no common agreement on the transport protocol to use. Consequently, the authors implemented their own reliable transport protocol. SNAIL [56] complemented this work by introducing SSL on an all IP architecture, leveraging the 6LoWPAN adaptation efforts done in the meantime. Together with the introduction of IP to the embedded world came the dilemma whether Transmission Control Protocol (TCP) is suited or not, due to its connection establishment overhead, poor performance in case of lossy networks and short term connections. For this reason, latest standardization efforts [144] assume User Datagram Protocol (UDP) at the transport layer, leaving reliability as an option to the application.

Unreliable transport and possible out of order delivery make TLS as is, an improper candidate for IoT. For the reason of securing application level protocols running over UDP in the Internet, such as Session Initiation Protocol (SIP), Real Time Protocol (RTP), or Media Gateway Control Protocol (MGCP), TLS has already been extended to Datagram Transport Layer Security (DTLS) [129, 107], which introduced additional 8 bytes of packet overhead in the form of the sequence and epoch numbers that were implicitly known with the reliable transport.

As a straightforward and standardized parallel to the successful model in the Internet, DTLS has attracted attention of the research community around the Internet of Things [128, 16, 82, 47, 127, 61, 18]. It is interesting to note, however,

that apart from the known advantage of using an already standardized protocol, no argument has been given on actual applicability of DTLS for IoT. Kothmayr *et al.* [82] discuss the necessity of authenticating both the client and the server during the DTLS handshake, but their experimental results show significant completion delays, ranging from 2 to 6.5 seconds. Granjal *et al.* [47] performed a comparative study on memory footprints, computational time, and required energy between IPsec protocols and DTLS, using different cryptographic suites. They showed similar performance of IPsec and DTLS, except when DTLS is additionally used to exchange keys with ECDH exchange. Kumar *et al.* [83] summarized DTLS memory requirements, communication overhead in terms of the number of messages, and code size for different cryptographic primitives.

Recognizing the excessive overhead of the DTLS handshake, Hummen *et al.* [61, 60] proposed different techniques to lower its impact on constrained devices—certificate pre-validation and handshake delegation to the "delegation server". Raza *et al.* tackled the same problem by proposing a 6LoWPAN DTLS compression scheme [128] to reduce packet overhead. This work has been integrated with CoAP and released in the open-source form [127].

The recent work of Capossele *et al.* [18] explores the idea of abstracting the DTLS handshake as a CoAP resource and implementing the handshake procedure using CoAP methods. The advantage of this approach is that DTLS can leverage the reliability of confirmable CoAP messages, as well as the blockwise transfer for large messages. The drawback, however, is lost backward compatibility with the existing Internet infrastructure.

A significant drawback of using DTLS to secure IoT is its incompatibility with multicast traffic. As stated by its designers [107], DTLS targets typical connection-oriented client-server architectures. While some of the IoT envisioned applications could loosely undergo this assumption, the majority cannot (see Chapter 8). In fact, group communication support is one of the main features why CoAP protocol has been standardized at all [144].

Additional concern raised by the straightforward, point-to-point use of DTLS is incompatibility with scenarios where the end-host in the Internet only supports HTTP/TLS. Brachmann *et al.* [16] discussed a possible DTLS/TLS mapping done at the gateway that preserves end-to-end security. While verifying integrity at the transport layer, however, it is impossible to perform the CoAP/HTTP mapping at the application layer, because DTLS will detect the alterations and drop the packets.

Gerdes *et al.* [43] leverage DTLS to piggyback authorization information to the constrained node. Such approach is interesting from the point of view of code size, as an existing DTLS implementation can be extended to provide authorization functionalities as well. Performance-wise and capability-wise, however, this approach inherits the drawbacks of DTLS discussed above.

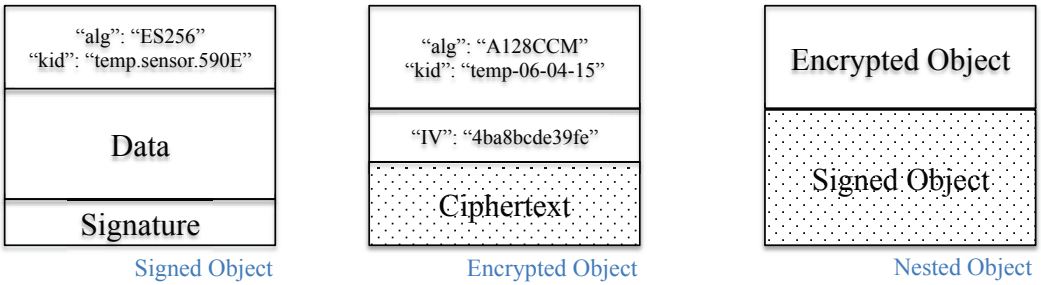


Figure 5.1: Examples of different secured objects.

5.3.4 Application Layer and Object Security Approaches

Object security is a generic term for cryptographically protected self-contained information object. Secured objects are typically used to wrap application data, such as electronic mails. An interesting property of secured objects is their statelessness – all the information necessary for their decryption/verification is communicated within the objects which makes object security interesting for secure data storage. For example, encrypted object in Fig. 5.1 is encrypted with a key whose identifier is "temp-06-04-15". Party that is interested in decrypting this object should have the key that corresponds to the "temp-06-04-15" identifier in order to be able to decrypt the object and access information contained within. If the key is not available locally, one could request the key from some online registry, such as the Authorization Server in charge of the domain that manages data access. We show some basic examples of *signed*, *encrypted*, and *signed-encrypted* objects in Fig. 5.1. As it can be inferred from the figure, different object security formats allow nesting of secured objects.

The benefits of object security were recognized and discussed as an option for securing the IoT [142, 25]. Sethi *et al.* [142] proposed an architecture relying on heavy utilization of public-key cryptography and Javascript Object Notation (JSON) Web Signatures, in order to facilitate the usage of intermediate proxies that can respond to requests with authentic data on behalf of constrained devices. Their work, however, does not aim to provide confidentiality and the authors hint that this could be achieved by performing a Diffie-Hellman (DH) exchange between a constrained device and a proxy server. Granjal *et al.* [48] design CoAP security options to facilitate the transport and signaling of the secured payload, over proxies and across different security domains. On the other hand, Seitz *et al.* leverage the benefits of object security to provide fine grained access control within an assertion-based authorization framework [140].

5.3.5 Standardization Efforts of IETF

In the following, we briefly survey main standardization activities within Internet Engineering Task Force (IETF), related to constrained devices and security.

IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH). 6TiSCH working group of IETF discusses a design of a security architecture to enable bootstrap-

ping of IEEE 802.15.4 nodes [149, 150, 131]. The main challenge is initial network access of devices that come from different vendors and therefore belong to different security domains. Devices are assumed to be pre-configured by their vendor with some cryptographic material, such as a PSK or a certificate. Once such a device is admitted into the network, it is necessary to have a standardized mechanism to setup link-layer keys and other parameters necessary for network operation.

Datagram Transport Layer Security (DTLS). One of the outcomes of CoAP standardization was to mandate the usage of DTLS, as a parallel to secure HTTP [144]. As we will see in Chapter 7, this was a heavy choice for networks where duty cycle must be kept low in order to preserve energy. DTLS In Constrained Environments (DICE) working group of IETF works on a recommended subset of DTLS features and extensions that can facilitate its deployment in IoT scenarios [164]. For instance, this DTLS profile recommends default retransmission timeout values to use, session resumption and keep-alive mechanisms in different IoT settings, as well as general recommendations on random number generation, among others.

Incompatibility of DTLS with multicast is bothersome in IoT scenarios. Some proposals within IETF aimed at extending DTLS with multicast support by reusing its record layer and relying on an independent group key management protocol [74]. These were quickly dismissed, as they revisit the core (D)TLS design assumption – point-to-point communication. A promising approach to reduce the communication overhead of the DTLS handshake is the session resumption without server-side state [135], and the latest version of the DTLS profile for constrained devices [164] recommends its use. One opposing proposal to CoAP being secured with DTLS was the establishment of security associations between two endpoints using CoAP options [180]. While the proposal leverages benefits of having security at the application layer, it essentially relies on the concept of a security handshake between two parties, rendering multicast communication unsupported.

Object security. There are several existing standards that specify object security format. Cryptographic Message Syntax (CMS) is a historic standard that is used to secure electronic mails with Secure/Multipurpose Internet Mail Extensions (S/MIME). Another, more recent, examples are formats based on JSON [66, 68, 65], that were standardized as part of the JSON Object Signing and Encryption (JOSE) working group. Both CMS that is based on Abstract Syntax Notation One (ASN.1) and JSON-based JOSE, introduce significant encoding overhead that is not acceptable for constrained devices and radio technologies with small Maximum Transmission Unit (MTU) sizes. For this reason, Concise Binary Object Representation (CBOR) objects are gaining popularity and in April 2015, a new working group was created to standardize object security format for constrained devices: CBOR Object Signing and Encryption (COSE).

Authorization. Authentication and Authorization for Constrained Environments (ACE) working group of IETF was created in June 2014 to standardize an authorization solution for IoT. First step in this process was the collection of requirements based on various IoT use-cases [139]. Requirements that are discussed

by ACE, however, seem to be contradictory with the initial choice of DTLS as a security protocol, particularly when it comes to proxies and caching. At the time of the writing, there are three principal approaches that are discussed for the ACE solution[4]:

- **OAuth**. Solution [165, 162, 163] based on the OAuth protocol [53], widely deployed in the Internet.
- **DTLS**. A solution called Delegated CoAP Authentication and Authorization Framework (DCAF) [44] based purely on DTLS.
- **Object security**. Solutions [141] [c_9] that leverage object security to protect against untrusted intermediaries (e.g. proxies) and support group communication.

5.4 Conclusion

There are several interesting points that can be taken from the surveyed work tackling different security challenges around IoT.

On one hand, link-layer security mechanisms represent the first line of defense to secure the local area network from outsiders and protect the users' privacy by encrypting the data exchanged on the radio channel. The reported performance costs of link-layer security in the literature are, however, contradictory and dependent on the evaluation methodology. For this reason, the first question that we answer in our research is the real-world cost of link-layer security mechanisms.

On the other hand, because IoT application data traverses multiple IoT and Internet hosts that are not trusted by the user, end-to-end security mechanisms are a necessary building block of IoT systems. The academic community has mostly focused on adapting the existing Internet end-to-end security standards to constrained devices. Reported studies consider the performance costs of an already-established security session and focus on optimizing the packet overheads by compression, making these mechanisms feasible, once the end-to-end session has been established. In parallel, developments in the standardization communities consider DTLS protocol as the de-facto solution for securing the IoT. However, DTLS cannot be used to meet all the IoT application requirements, due to the protocol-level design incompatibilities with CoAP.

The applicability of DTLS to IoT environments leads us to another two research questions. To understand the costs of session establishment, the second question addressed in our research is the performance cost of DTLS handshake in energy-constrained environments, where network nodes are duty-cycled. As the third research question, we explore whether it is possible to secure IoT applications without posing requirements on the communication paradigms from the traditional Internet, as done when DTLS is used.

[4]Latest developments in ACE, as of the meeting in Yokohama on Monday, November 2nd, 2015, consider object security as a building block of all proposed solutions.

The remaining of this Part details the answers to the three research questions: 1) In Chapter 6, we evaluate the cost of link-layer security in terms of network performance for wireless networks based on IEEE 802.15.4 standard; 2) In Chapter 7, we study the establishment of a DTLS handshake with the overviewed duty cycling mechanisms; 3) In Chapter 8, we leverage object security to design OSCAR, a network security architecture that is compatible with caching at untrusted intermediaries and group communication, while being backwards compatible with existing deployments based on DTLS.

Chapter 6

Wireless Network Security and its Overhead

Link-layer security mechanisms are tightly bound to the underlying radio technology. Our study in this chapter focuses on IEEE 802.15.4 networks. We aim at answering the following question: How much does security degrade network performance?

We give an overview of the main IEEE 802.15.4 security features in Section 6.1 in order to study their effect on network performance. On one hand, in *beacon-enabled* mode of IEEE 802.15.4 that we use on GreenNet nodes, time latencies are not critical. However, as with any harvested system, energy is of utmost importance. We therefore study the energy overhead of IEEE 802.15.4 security processing on GreenNet nodes in Section 6.2 and compare that to contradictory conclusions in the literature. On the other hand, Time-Slotted Channel Hopping (TSCH) mode of IEEE 802.15.4e has an important dependency on security execution time. Minimal TSCH slot size that affects throughput and network delay is tightly bound to security processing. This dependency is the main topic of our study in Section 6.3.

6.1 Security in IEEE802.15.4 networks

Security extension of the IEEE 802.15.4 standard addresses the typical privacy and security concerns discussed in Chapter 5. The standard completely relies on sym-

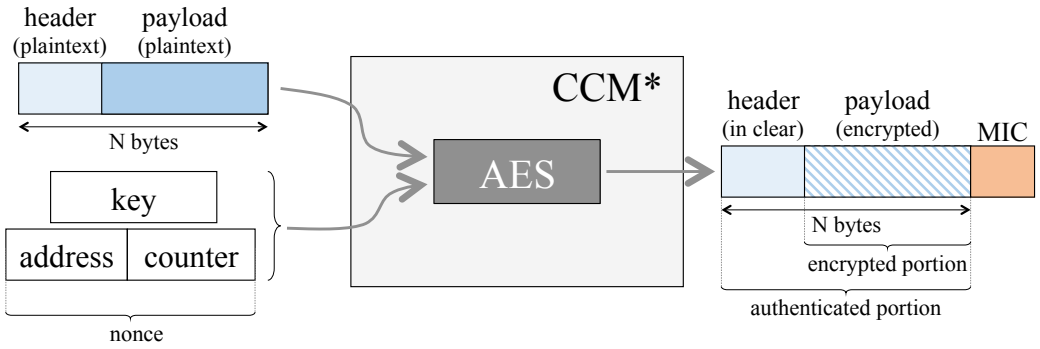


Figure 6.1: IEEE 802.15.4 uses Extension of Counter Mode Encryption and Cipher Block Chaining Message Authentication Code (CCM*) to secure radio frames. Nonce is created from the address of the sender and monotonic counter to avoid its reuse and to protect from replay attacks.

Table 6.1: The security levels in IEEE 802.15.4.

Level	Encrypted payload	MIC length
MIC-32	NO	4 bytes
MIC-64	NO	8 bytes
MIC-128	NO	16 bytes
ENC	YES	*no MIC*
ENC-MIC-32	YES	4 bytes
ENC-MIC-64	YES	8 bytes
ENC-MIC-128	YES	16 bytes

metric cryptographic primitives and it uses CCM* mode to provide confidentiality, data origin authentication and protection from replay attacks on a per frame basis. The standard delegates host authentication and key exchange functions to upper layers of the network stack.

As we saw in Chapter 4, CCM* is a "wrapper" cryptographic primitive around Advanced Encryption Standard (AES) that uses Counter (CTR) mode for encryption and Cipher Block Chaining Message Authentication Code (CBC-MAC) for authentication. It encrypts and/or authenticates an arbitrarily long sequence of plaintext bytes. When applied to a link-layer frame, this means that CCM* can encrypt the Medium Access Control (MAC) payload while keeping the MAC header intact. When used to authenticate a frame, CCM* calculates a Message Integrity Code (MIC) over the complete frame. This MIC is truncated to the desired length (4, 8 or 16 bytes), and appended at the end of the frame.

Each frame secured with a given key must use a different nonce. Encrypting two frames with the same nonce has severe consequences on security: plaintext of both frames may be easily recovered. By constructing the nonce with a monotonic counter, it is possible to ensure replay protection for two communicating nodes. We depict this in Fig. 6.1.

Table 6.1 presents the security levels of IEEE 802.15.4. Levels differ in MIC length and whether encryption is applied on the payload or not. A longer MIC provides higher security as the probability of blind forgery by guessing the code is lower – 2^{-32}, 2^{-64}, 2^{-128} for 4, 8 or 16 bytes MIC, respectively [136]. A higher security level induces a larger frame due to the longer MIC, but computational overhead stays the same. Larger frames can be a concern since IEEE 802.15.4 Maximum Transmission Unit (MTU) is only 127 bytes so longer MIC translates to a reduction in the available payload size. Each frame can use a different security level. Local policies dictate if the security level of the received frame should be accepted or not. These conformance checks are a pre-requisite for the CCM* verification to be invoked.

Fig. 6.2 illustrates the frame format with security features enabled. Each secured frame carries an Auxiliary Security Header (ASH) with signaling information related to the key and nonce. In IEEE 802.15.4-2011, the nonce is created from the address

of the sender and the local frame counter that increments for each transmitted frame. The 4-byte frame counter must be signaled to the recipient, and is therefore included in ASH. Because the recipient keeps track of the last frame counter it received from a given neighbor, frames are protected against replay attacks. Key signaling overhead varies with the key management scheme used, and can range from 0 bytes for implicit keying to 9 bytes. The total ASH overhead (not including the MIC) ranges from 5 to 14 bytes.

The upper layer sets the specific security level and the key to be used on beacon, command or data frames. However, Acknowledgment (ACK) frames in IEEE 802.15.4-2011 do not support security and are always sent in clear.

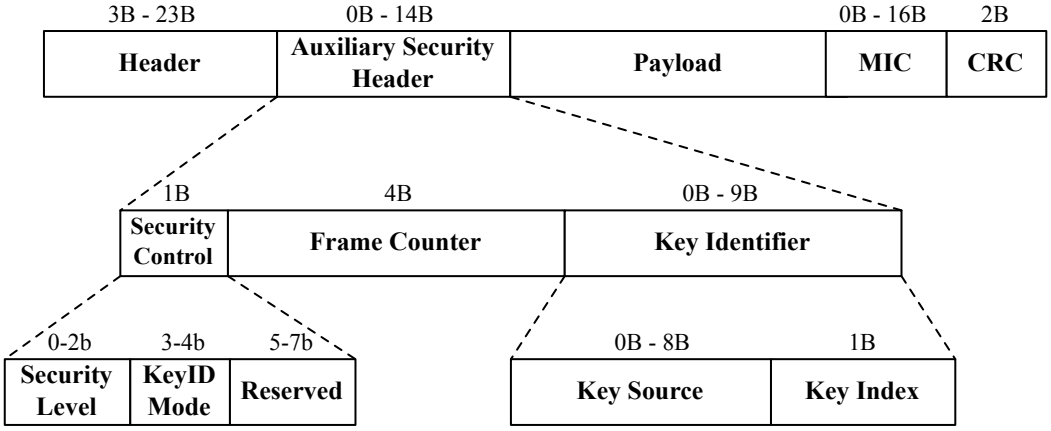


Figure 6.2: IEEE 802.15.4 frame format with security enabled. Reserved bits are used for TSCH-related signaling.

6.2 Effect on Energy Consumption with IEEE 802.15.4 Beacon-Enabled Mode

Although *beacon-enabled* mode of IEEE 802.15.4 requires network-wide synchronization, latency introduced by security operations is not a critical aspect. Rather, we are interested in energy penalty that security brings along [c_6]. One of our goals is to revisit the results and conclusions published in the literature [109, 28] on the overall cost of security that ranges up to 91%.

We implemented IEEE 802.15.4 security features for GreenNet nodes and *beacon-enabled* mode. We leverage the available AES hardware accelerator. Similarly to the crypto engine implementation presented in Chapter 4, we use hardware implementations of CTR and Cipher Block Chaining (CBC) modes, and build a CCM* implementation on top[1].

We implemented and tested all the security levels specified in Table 6.1 but present results for the three modes where both confidentiality and integrity of data

[1]Nodes were clocked at 12MHz in these experiments.

is desired. It is important to stress that apart from the CCM* cryptographic processing, an IEEE 802.15.4 security extension details verification procedures for incoming and outgoing frames. These procedures protect from protocol level attacks on IEEE 802.15.4 nodes and in terms of processing can take as much as hardware-accelerated crypto overhead. Fig. 6.3 presents the testbench in Crolles, France where we performed the experiments.

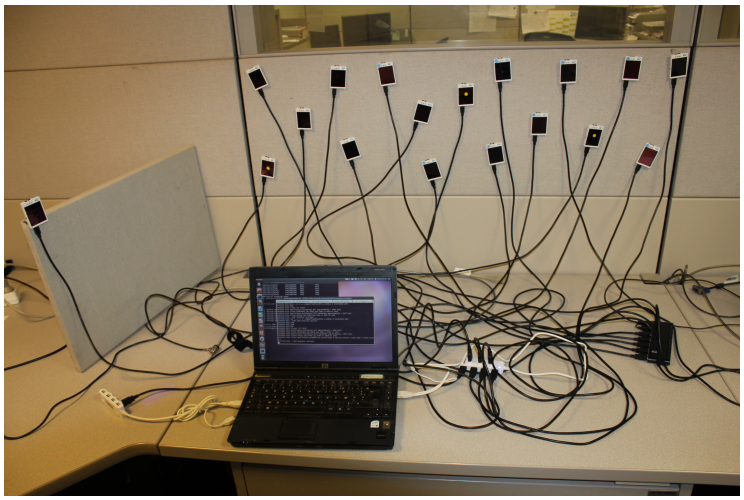


Figure 6.3: Testbench with 18 energy-harvested GreenNet nodes. Nodes are connected to Universal Serial Bus (USB) for the collection of experiment traces.

6.2.1 Methodology

In our experiments, we study the performance degradation in terms of energy for a GreenNet leaf node sending temperature readings with a period of 4 minutes. Therefore, a device spends most of the time sleeping and is woken up by an interrupt just before the beacon preceding the desired application interval. Consequently, the device receives the beacon, synchronizes with the rest of the network and transmits the sensor reading using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) algorithm. Sensor reading fits within one IEEE 802.15.4 radio frame and there is no fragmentation even with the highest security level and 16-bytes long MIC.

We estimate the average power consumption of a leaf node over a sufficiently long period (6 hours) to derive conclusions regarding the overall effect of security features on autonomy and lifetime of a GreenNet node.

We estimate energy consumption using Energest, a Contiki per component profiling tool [35]. Energest measures the time spent by different components on a platform in a given state (for instance, the time Central Processing Unit (CPU) spent in active or low power mode; radio transceiver in receive or transmit). These values are converted to energy by multiplying with the constant operating voltage (we used 3V) and the current draw values from appropriate data sheets. All the

Table 6.2: Experiment setup.

Settings	Value
Operating System	Contiki OS
Application	Temperature sending, period of 4 min
Network stack	CoAP + UDP + IPv6 + 6LoWPAN
MAC and Radio Duty-Cycling (RDC)	beacon-enabled IEEE 802.15.4
Beacon Interval	492 ms
Superframe Duration	31 ms
TX power	0 dBm
RF channel	13 and 19 (2.415 MHz and 2.445 MHz)
Experiment duration	6 h

points in the following graphs were averaged over 10 experiment runs, each lasting 6 hours. We detail the experiment setup in Table 6.2.

During its lifetime, a GreenNet node undergoes two distinct phases: 1) bootstrapping phase; 2) steady state operation according to application requirements. In phase 1, node needs to discover its environment by passively scanning the radio channel, to associate (link-layer operation) to the nearest coordinator in the network and register its application resources to the network gateway. In our deployment, node registers to the network gateway using the ZigBee Smart Energy Profile (SEP) specification [184]. During this initial phase, node performs extensive application level negotiation with the network gateway, resulting in high traffic load and occasionally large application payload. Once the registration is complete, the node starts to follow its application schedule and wakes up every 4 minutes to send its current temperature reading. In the evaluations, we distinguish the two phases.

6.2.2 Energy Consumption in Steady State

We first evaluate the main contributors to the overall energy consumption in steady state (see Table 6.3). We focus on 4 main components: 1) energy spent due to sleep mode leakages of the Microcontroller Unit (MCU) and the rest of the board – Low Power Mode (LPM); 2) energy spent due to active CPU computations; 3) energy spent while the radio transceiver is in receive mode (RX); 4) energy spent while the radio transceiver is in transmit mode (TX).

We may note in Table 6.3 that without security, as much as 69% of energy is spent for sleep mode leakages. An LPM drop observed in Table 6.3 with increasing security level is due to the overhead introduced by other components which effectively results in less time spent sleeping. Therefore, in terms of energy consumption, security may have an effect on the components that roughly make up one third of the overall consumption. We can note in Table 6.3 that the percentage of energy spent on radio communication increases with the security level. A higher security level corresponds to a longer MIC and a larger byte overhead, which directly affects

the radio transmission/reception cost. Cryptographic processing causes a sharp increase in CPU usage from `No Security` to `ENC-MIC-32` modes, while there is a slight increase among different security levels. CCM* algorithm has constant computational complexity, independent of MIC length and the security level but longer radio operations prolong the time MCU stays active.

Table 6.3: Steady-state contributors to energy consumption of GreenNet temperature sensor.

Security Level	Low Power Mode	CPU active	Radio RX	Radio TX
`No Security`	69.77%	15.77%	10.92%	3.54%
`ENC-MIC-32`	68.57%	16.35%	11.30%	3.78%
`ENC-MIC-64`	67.62%	16.63%	11.69%	4.06%
`ENC-MIC-128`	67.11%	16.80%	11.81%	4.28%

Fig. 6.4 presents the average power consumption measured over the experiment duration. It is interesting to note in Fig. 6.4 the amount of power drawn while the radio transceiver is in receive mode. Although the application executing during the experiments induced typical convergecast traffic, with no packets going in the downward direction to the nodes, power drawn while receiving is 3 times higher compared to power drawn while transmitting. Our node sleeps most of the time, but just before transmitting the sensor reading, it needs to resynchronize to the rest of the network by receiving a beacon frame at a precise interval. Delays caused by hardware imperfections, such as crystal oscillator startup delay, calibrations, clock drifts, and similar, induce margins that are an important component in the overall power consumption. Roughly 15% of the receive mode consumption comes from the actual physical reception of a beacon frame, whose size increases with the security level. As a consequence, 85% of receive mode consumption is due to idle listening and is security agnostic. Thus, hardware idealizations, as often done in the literature, can result in misleading results.

We summarize in Fig. 6.5 the overall energetic cost of security for GreenNet nodes. A clear effect of larger frames due to the larger MIC can be noted for the radio transmit mode as its consumption is caused by physical transmission. Fig. 6.5 might be misleading in terms of the consumption in active CPU mode, as we can observe an increase with the security level, although CCM* has constant computational complexity. MCU, however, stays in active mode while (longer) frames are received or transmitted by the radio transceiver. This introduces a correlation of the nominal energetic cost of CPU computations with increasing security levels.

Overall cost of IEEE 802.15.4 security in our scenario ranges from 1.75% to 3.96%, depending on the security level. For energy-harvested platforms, such as

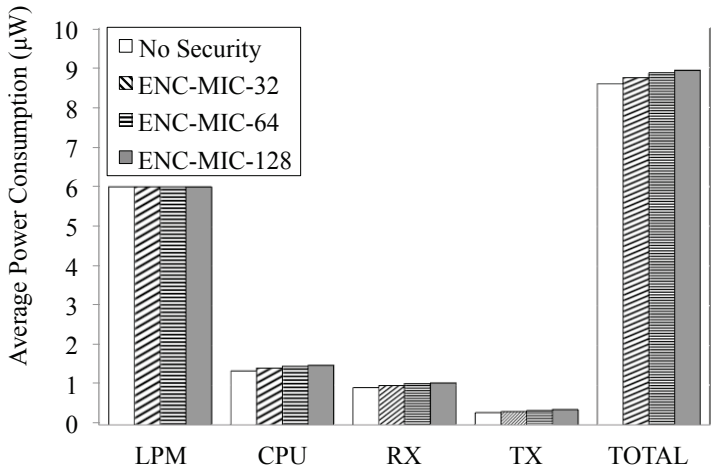


Figure 6.4: Average power consumption of GreenNet temperature sensor over 6 hours.

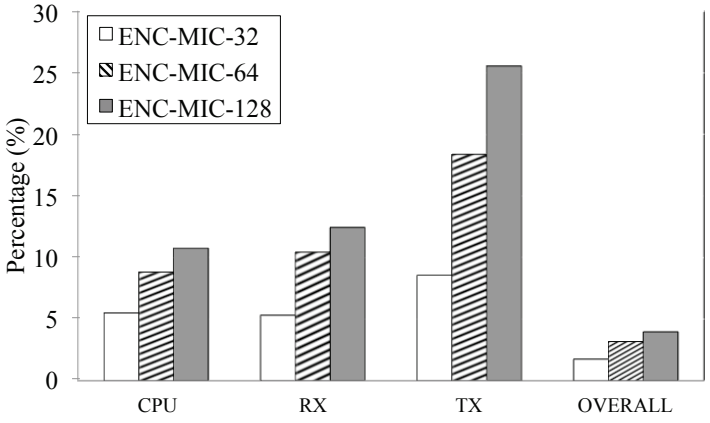


Figure 6.5: Energetic cost of IEEE 802.15.4 security for GreenNet temperature sensor.

GreenNet, this result directly corresponds to the requirement that 1.75% to 3.96% extra energy needs to be harvested from the environment, in respect to the scenario without security.

6.2.3 Application Registration Phase

ZigBee SEP [184] defines an application level negotiation protocol for the initial resource subscription and discovery. In our deployment, we use an optimized version over Constrained Application Protocol (CoAP). This initial negotiation phase results in extensive communication (i.e. high traffic load) between a node and the network gateway.

Table 6.4: Average power consumption during the application registration phase (i.e. bootstrapping) using SEP.

Security Level	Average Power Consumption [µW]	Cost
`No Security`	733.76	
`ENC-MIC-32`	777.66	5.98%
`ENC-MIC-64`	802.92	9.43%
`ENC-MIC-128`	907.93	23.74%

Table 6.4 presents the average power consumption for different security modes measured during the negotiation phase. First of all, we can notice higher power consumption, even without security, as nodes do not spend as much time sleeping but wake up to receive a beacon each 0.48s interval. They engage in bidirectional communication with the gateway over a period of approximately 10s.

The lower two security levels (`ENC-MIC-32` and `ENC-MIC-64`) cause a moderate increase in consumption which is to be expected with the increased communication load. However, the highest security level `ENC-MIC-128` introduces a significant, 23.74% increase. The explanation is fairly simple and is due to fragmentation. With the additional security overhead, some SEP packets exceed the IEEE 802.15.4 MTU limit and therefore get fragmented. As there were no contending transmissions with CSMA/CA, fragmented frames only slightly affect the SEP phase duration. Average power consumption, on the other hand, sees a large increase.

6.3 Effect on TSCH Slot Duration

TSCH mode of IEEE 802.15.4e imposes strict constraints due to the precise timings within a timeslot. We are interested in quantifying the effect of security processing

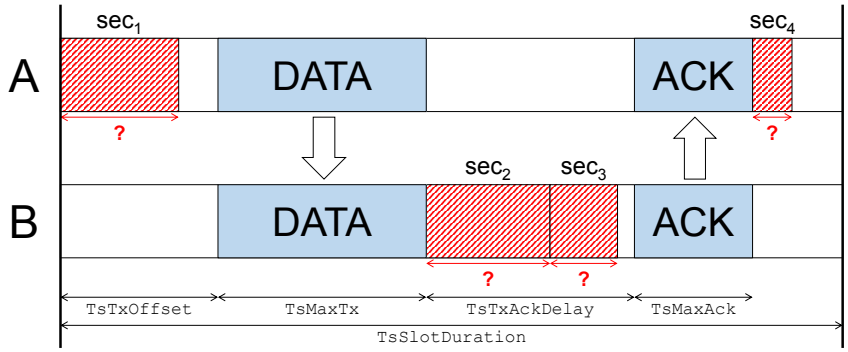


Figure 6.6: The minimal duration of a timeslot in an IEEE 802.15.4e TSCH network depends on how long link-layer security operations take.

on TSCH timeslot length[2] [c_3]. The shorter the timeslot, the lower the latency and the higher the throughput of a network. Yet, how short a timeslot is can depend on the time it takes to secure/unsecure frames, which involves encrypting/decrypting and/or authenticating with CCM*. Different platforms have different hardware capabilities: on some, link-layer security is implemented in hardware; on others, these operations need to be done in software.

Fig. 6.6 depicts the operations that occur in a timeslot when `A` sends a data frame to `B`:

- `A` secures the data frame following IEEE 802.15.4 outgoing security procedure and CCM*. We call this operation sec_1.
- At precisely `TsTxOffset` into the timeslot, `A` sends this (secured) frame to `B`. The transmission of the data frame takes at most `TsMaxTx`.
- `B` unsecures the data frame, following incoming security procedure (operation sec_2).
- If the unsecuring operation is successful, `B` secures an ACK frame (operation sec_3).
- Exactly `TsTxAckDelay` after receiving the end of the data frame, `B` sends the (secured) ACK frame. The transmission of the ACK frame takes at most `TsMaxAck`.
- `A` unsecures the ACK frame (operation sec_4). If successful, it removes the data frame from its transmission queue.

Eq. (6.1) indicates the timing constraints the duration of the security operations puts on the different TSCH timings. We denote $dur(sec_1)$ the duration of sec_1.

[2]Experimentation results presented in this section were obtained by Savio Sciancalepore at Politecnico di Bari.

$$\begin{cases} \texttt{TsTxOffset} \geq & dur(sec_1) \\ \texttt{TsTxAckDelay} \geq & dur(sec_2) + dur(sec_3) \\ \texttt{TsSlotDuration} \geq & \texttt{TsTxOffset} + \texttt{TsMaxTx} + \\ & \texttt{TsTxAckDelay} + \texttt{TsMaxAck} + \\ & dur(sec_4) \end{cases} \tag{6.1}$$

6.3.1 Security Additions Introduced by IEEE 802.15.4e and TSCH

There are several important points that differ between IEEE 802.15.4 and IEEE 802.15.4e in terms of security which are important for understanding the following results. IEEE 802.15.4e introduces Information Elements (IE) to exchange information between neighbor nodes in TSCH networks. Nodes maintain synchronization by indicating the time correction as part of an IE in ACK frames. An attacker could perform Denial of Service (DoS) attacks by altering this time correction; for this reason IEEE 802.15.4e added support for secured ACK frames.

Time synchronization in the network means that all nodes share the Absolute Slot Number (ASN): the number of slots which have passed since the network has started. The ASN is forever incrementing[3]. A common notion of time simplifies replay protection as a node does not need to maintain a frame counter for each of its neighbors. Instead, TSCH uses the ASN as a frame counter and omits its inclusion in the ASH, reducing the overhead by 4 bytes.

The use of ASN in the nonce implies that the sec_1 operation can only be done once the ASN of the slot is known (see Fig. 6.6). In practice, this means that pre-calculating the sec_1 operation is not possible. Operation sec_1 includes the key lookup and CCM* encryption on a potentially maximum length frame (127 bytes). Before the receiving node can transmit an ACK, it must verify the conformance of the frame against local security policies and decrypt/authenticate it (sec_2). Finally, the node prepares and secures the ACK frame (sec_3), which includes the time correction indication. Before the time correction can be applied on the transmitted side (node `A` in Fig. 6.6), the ACK frame must pass the CCM* check and conformance verifications (sec_4). The duration of the sec_1, sec_2, sec_3 and sec_4 operations on different hardware platforms directly influences the minimum slot duration, which we evaluate experimentally.

6.3.2 Methodology

Our goal is to quantify the overhead of link-layer security on different hardware and using different CCM* implementation strategies[4]. We use TelosB board as an example platform that has MCU and radio connected over Serial Peripheral

[3]The ASN is encoded on 5 bytes. With a 10 ms timeslot duration, the ASN value rolls over every 350 years.

[4]Following results leverage CCM* implementations that are independent of those presented in Chapter 4.

Table 6.5: Implementation Strategies.

strategy	AES	CCM*
"software"	in software	in software
"hybrid"	in hardware	in software
"hardware"	in hardware	in hardware

Interface (SPI), and OpenMote-CC2538 as an example board based on a System on Chip (SoC) architecture (see Chapter 2).

We use the OpenWSN implementation on both platforms. OpenWSN protocol stack includes IEEE 802.15.4e-2012 TSCH, IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN), IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) and CoAP [177]. We augment OpenWSN with link-layer security, implemented using three strategies: in software, in hardware, and a hybrid solution. As shown in Table 6.5, these strategies differ in whether they exploit hardware acceleration for AES and/or CCM*.

The "software" implementation strategy consists in implementing both AES and CCM* in software. The "hardware" implementation strategy consists in exploiting hardware acceleration for both AES and CCM*. As we could see in Chapter 4, many solutions offer AES hardware acceleration, but not CCM*. To measure the overhead of link-layer security on those platforms, we adopt a "hybrid" implementation strategy in which we rely on hardware for AES, but implement CCM* in software. The hybrid implementation hence uses hardware-accelerated AES block cipher. The rest of the CCM* algorithm, which includes CTR and CBC-MAC modes of operation, creation of plaintext and ciphertext is handled through software instructions.

We evaluate minimum obtainable timeslot duration given the duration of the link-layer security operations that include CCM*. The experimental setup consists of two nodes forming one network [c_3]. One of the nodes is the root of the network, the other is a leaf node that attaches to the root. In each case, after loading the appropriate software on the nodes, we boot the root node and wait for the leaf node to synchronize to it. The root node is attached to a computer; from that computer we use the `ping` program to verify connectivity to the leaf node. `ping` allows us to choose the size of the payload sent in the ICMPv6 echo request/response packets; we choose it so that the resulting link-layer frame is always 127 bytes long (the maximum length for IEEE 802.15.4-compliant nodes).

In OpenWSN, IEEE 802.15.4e TSCH is implemented as a finite state machine. Different timings, illustrated in Fig. 6.6, cause the state machine to advance. `TsTxOffset` is an example timing: when it expires, the state machine kicks off the transmission of the frame. This means, at that point in the timeslot, all the operations for preparing the packet (including sec_1) need to be complete. The OpenWSN implementation is instrumented so that the different timings can be "read" on a set of digital pins using a logic analyzer. This allows us to measure the duration of the sec_1, sec_2, sec_3 and sec_4 operations, and verify the `TsTxOffset`, `TsTxAckDelay` and `TsSlotDuration` durations. For a detailed overview of the measurement procedure,

the reader should refer to Sciancalepore *et al.* [c_3].

6.3.3 Minimal Slot Length

Figs. 6.7 and 6.8 present the minimum obtained `TsTxOffset`, `TsTxAckDelay` and `TsSlotDuration` durations, for the TelosB and OpenMote-CC2538 platforms, respectively. The hardware implementation strategy results in a shorter slotframe than the software implementation. Depending on the security level, a hardware-based implementation of link-layer security will result in a timeslot duration reduced by a factor of 3 to 4.

The difference in timeslot duration between "software" and "hybrid" implementation strategies reflects the advantage of having AES execute in hardware. Similarly, the comparison between "hybrid" and "hardware" implementations reflects the advantage of a hardware-based CCM*.

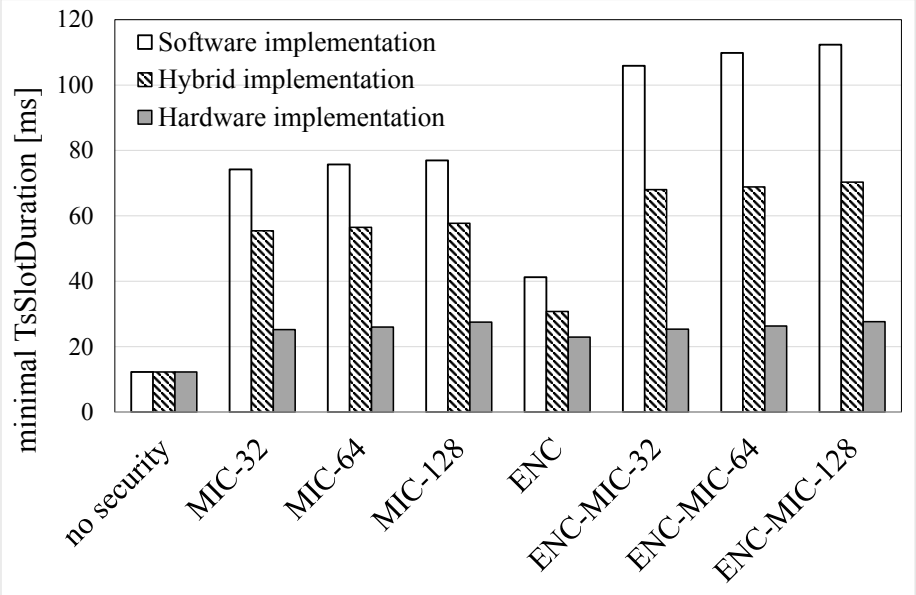
On slower platforms such as the TelosB (Fig. 6.7), the biggest gains are made by running AES in hardware.

The most common security level in TSCH networks (including WirelessHART and IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH)) is `ENC-MIC-32`, i.e. frames are encrypted and a 4-byte MIC is used for authentication. We highlight that security level in Figs. 6.7 and 6.8. A full software implementation on an older platform such as the TelosB results in a minimal timeslot duration of 106 ms. Using hardware acceleration reduces the time duration by a factor of 4, down to 25 ms. Switching to a state-of-the-art platforms, which features both faster hardware implementation of AES and Counter Mode Encryption and Cipher Block Chaining Message Authentication Code (CCM)*, and a single-chip architecture, allows the timeslot duration to be further reduced by a factor of 2, down to 12 ms[5].

6.4 Conclusion

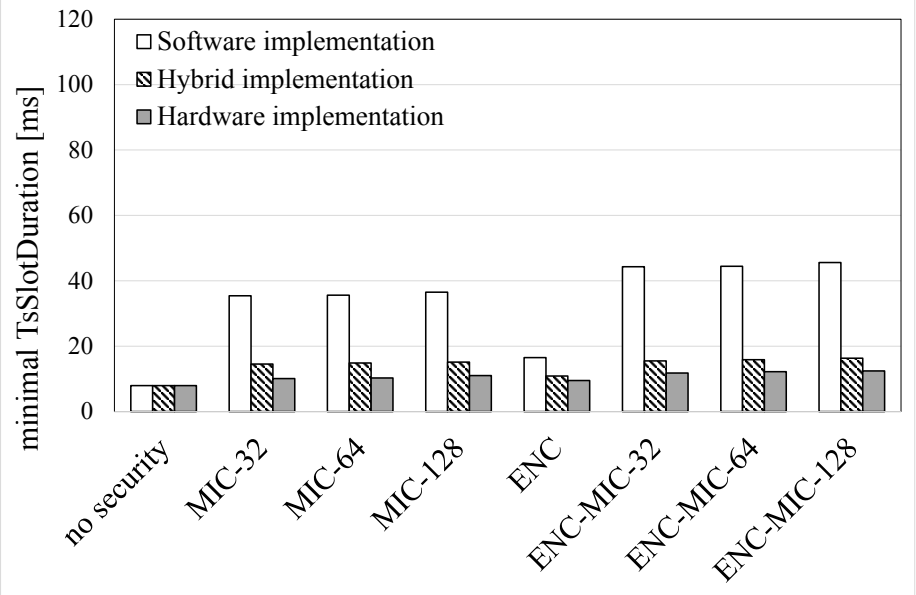
Results of performance evaluation from Sections 6.2 and 6.3 render some common concluding points. The TSCH case study with stringent timing requirements made it clear that hardware acceleration of cryptography is also necessary from the latency perspective. With GreenNet nodes as an example, we could see that link-layer security and symmetric-key primitives introduce a negligible overhead in terms of energy, as long as the fragmentation threshold is not reached. Both cases studies (beacon-enabled and TSCH) indicate that performance degradation of link-layer security is acceptably low and overweighted by the added-value of services it brings along. We therefore conclude that future challenges in this context are mostly related to implementation-level optimizations and tradeoffs that our CRYPTO ENGINE Application Programming Interface (API) from Chapter 4 tries to tackle. For our

[5]Latest results that leverage more optimized implementations of IEEE 802.15.4 security procedures and CCM* indicate that OpenMote-CC2538, as well as GreenNet boards using `ENC-MIC-32` level can use timeslot duration bellow 10ms.



Security Level	Software Implementation			Hybrid Implementation			Hardware Implementation		
	TsTx Offset [ms]	TsTx AckDelay [ms]	*minimal* TsSlot Duration [ms]	TsTx Offset [ms]	TsTx AckDelay [ms]	*minimal* TsSlot Duration [ms]	TsTx Offset [ms]	TsTx AckDelay [ms]	*minimal* TsSlot Duration [ms]
no security	3.63	4.46	12.24	3.63	4.46	12.24	3.63	4.46	12.24
MIC-32	26.70	42.48	74	18.89	31.40	55.42	7.75	11.87	25.21
MIC-64	27.31	42.76	75.71	19.10	31.77	56.49	7.81	12.18	25.97
MIC-128	27.71	43.37	76.94	19.41	32.47	57.71	7.90	12.79	27.50
ENC	15.93	16.88	41.23	12.60	13.37	30.76	7.66	10.01	22.92
ENC-MIC-32	48.46	51.09	105.87	25.91	37.05	68.02	7.78	12.48	25.36
ENC-MIC-64	50.23	51.39	109.83	25.97	37.42	68.85	7.83	12.54	26.28
ENC-MIC-128	52.52	51.91	112.34	26.28	37.96	70.31	7.97	12.94	27.65

Figure 6.7: Minimum timeslot duration for the TelosB mote [c_3].



Security Level	Software Implementation			Hybrid Implementation			Hardware Implementation		
	TsTx Offset [ms]	TsTx AckDelay [ms]	*minimal* TsSlot Duration [ms]	TsTx Offset [ms]	TsTx AckDelay [ms]	*minimal* TsSlot Duration [ms]	TsTx Offset [ms]	TsTx AckDelay [ms]	*minimal* TsSlot Duration [ms]
no security	2.47	0.95	7.93	2.47	0.95	7.93	2.47	0.95	7.93
MIC-32	10.41	14.07	35.00	3.84	3.30	14.53	2.62	1.19	10.10
MIC-64	10.44	14.19	35.58	4.00	3.42	14.83	2.69	1.28	10.28
MIC-128	10.47	14.31	36.50	4.15	3.51	15.14	2.72	1.34	11.01
ENC	6.44	7.20	16.51	3.23	2.01	10.86	2.59	0.98	9.49
ENC-MIC-32	14.37	18.80	44.28	4.24	3.88	15.50	3.51	1.23	11.78
ENC-MIC-64	14.50	18.86	44.43	4.27	4.00	15.84	3.54	1.31	12.21
ENC-MIC-128	14.53	19.26	45.60	4.30	4.30	16.33	3.69	1.40	12.45

Figure 6.8: Minimum timeslot duration for the OpenMote-CC2538 mote [c_3].

study, we next consider the problem of end-to-end security, where any intermediate node on the path of a message may be compromised.

Chapter 7

End-to-End Security with (D)TLS

Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) [30] are fundamental protocols supporting secure communication over the Internet. With the advent of Internet of Things (IoT) and its applications, we face the problem of supporting communication security for IoT devices that present stringent energy, memory, and CPU constraints. Datagram Transport Layer Security (DTLS) [129] is a version of TLS running over User Datagram Protocol (UDP) used by the Constrained Application Protocol (CoAP) [144], ZigBee IP and Lightweight Machine to Machine (LWM2M) (other standards for constrained IoT networks) to secure IoT application traffic.

Yet, running DTLS on constrained IoT devices is challenging, in particular because of the amount of traffic needed to establish a DTLS session and due to the memory footprint of a DTLS implementation [60]. DTLS benchmarks exist [47, 127] and focus on memory footprint and message size for different cipher suites.

IoT devices follow a sleep/wakeup schedule to minimize the time their radio transceivers are on, which reduces the energy consumption. Duty cycling results in higher latency and lower throughput, which has a direct impact on the DTLS performance. The goal of this chapter is to provide a thorough evaluation of the DTLS performance on top of representative duty-cycled networks. More specifically, our contributions are the following:

- We measure the duration of the DTLS handshake and energy consumption for the following three duty cycling protocols: preamble sampling [17], IEEE 802.15.4 beacon-enabled mode [62], and IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) [63]. We use several evaluation methods (emulation, measurements on a real-world testbed, and an analysis) to obtain meaningful results. This part is the core of the chapter.
- We quantify the impact of the limited memory on the number of simultaneous DTLS sessions a constrained IoT device can maintain.
- We show that the probability for a DTLS session establishment to fail because the server runs out of memory to hold the associated state can be modeled with the Engset loss formula.

The remainder of this chapter is organized as follows. Section 7.1 gives an overview of the DTLS protocol. Sections 7.2.1, 7.2.2 and 7.2.3 present the experimental and analytical performance results of DTLS on top of duty cycling protocols.

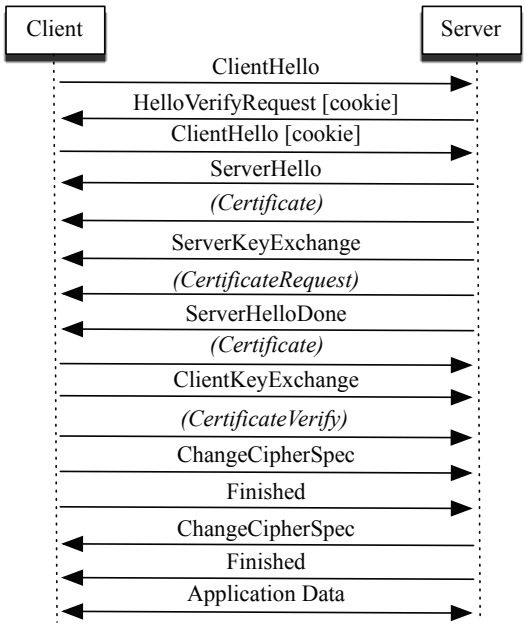


Figure 7.1: Message exchange during a DTLS handshake. Messages in parentheses are not sent for pre-shared key cipher suites.

Section 7.3 discusses the impact of memory constraints on DTLS. Section 7.4 concludes the chapter.

7.1 Datagram Transport Layer Security

Securing application traffic is often achieved by transferring data over a secure channel between the two communicating end-points. In the network stack, this secure end-to-end channel can be established at the network layer (e.g. IPsec), the transport layer (e.g. TLS), or the application layer (e.g. Secure Shell (SSH)). For application development, security at the transport layer is the most common. The *de-facto* security standard for the Internet application traffic is SSL and its Internet Engineering Task Force (IETF) successor TLS [30]. TLS was designed for client-server applications that operate over a reliable transport. To establish a secure channel, a client and a server first perform the TLS handshake during which they authenticate each other and derive the symmetric keys to use during the session.

DTLS [129] is an extension of TLS for datagram transport and runs over UDP rather than Transmission Control Protocol (TCP). Like TLS, DTLS protects the payload with encryption and authentication. DTLS records are 8 bytes longer than in TLS, as DTLS adds an explicit 8-byte sequence number. Stream ciphers, such as Rivest Cipher 4 (RC4), create an inter-record cryptographic context that introduces vulnerabilities with dropped and reordered packets. Consequently, DTLS bans the use of stream ciphers and relies on block ciphers for encrypting and authenticating records.

All messages carried by DTLS are encapsulated within DTLS *records* that add

a constant 13 byte overhead per datagram. The Record Layer supports four DTLS upper protocols: 1) *Handshake* protocol establishing a secure authenticated session between two peers, negotiating algorithms, and the key material; 2) *Alert* protocol signaling session closure or eventual errors; 3) *Change Cipher Spec* protocol signaling modifications to encryption strategies; and 4) *Application Data* protocol carrying application data. To deal with Denial of Service (DoS) attacks, the Handshake protocol uses a stateless *cookie* exchange: before the server allocates any resources, the client needs to resend the *cookie* thus proving that the client can receive messages at a given Internet Protocol (IP) address.

Fig. 7.1 shows a message exchange during a DTLS handshake. Once the client has replayed the stateless *cookie* from the server `HelloVerifyRequest` message, the server allocates the necessary resources. It chooses its preferred cipher from the client cipher set and notifies the client using a `ServerHello` message. The messages exchanged during key negotiation depend on the cipher. When using a pre-shared key, the message containing the server certificate is omitted and the server optionally sends `ServerKeyExchange` indicating to the client which pre-shared key to use. In this case, the two parties authenticate each other with the common secret, established out-of-band (also used to derive the session keys).

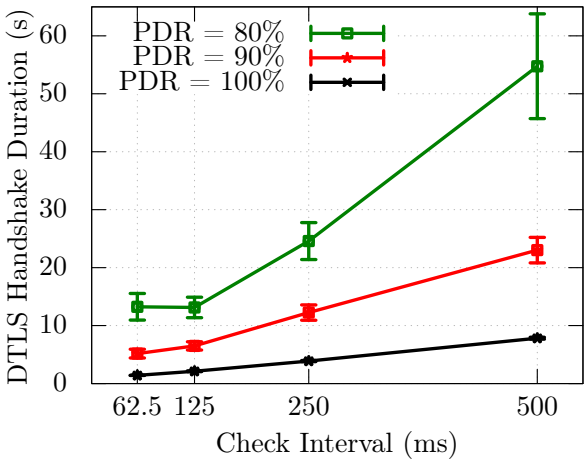
Because of different application types, DTLS has been used differently in IoT networks and on the traditional Internet. It is very common for a regular Internet host to establish short-lived TLS sessions, for example when browsing the Internet `https` Uniform Resource Locator (URL). An IoT device typically periodically reports sensor readings to a server and therefore uses one long-lived DTLS session, which is a good thing as constrained IoT networks cannot handle frequent expensive DTLS handshakes, as highlighted in this chapter. However, we are witnessing the emergence of applications in which mobile workers establish short-lived DTLS sessions with individual nodes using hand-held devices, for example for maintenance or drive-by metering [139]. In this context, it is important to understand the limitations of DTLS when duty cycling protocols are used.

In the following sections, we study the effects of duty cycling protocols on the performance of DTLS through emulation, real-world experimentation, and analysis.

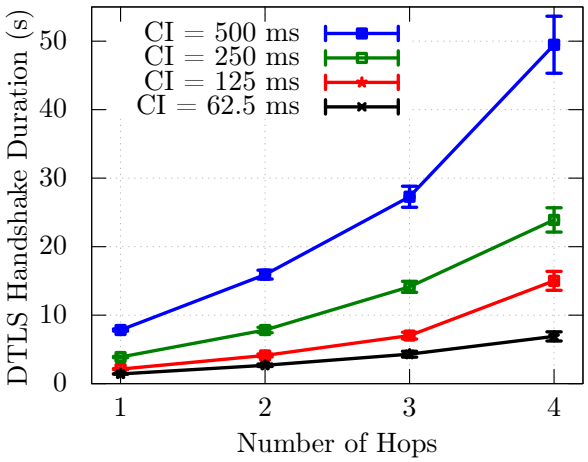
7.2 DTLS Performance in Duty-Cycled Networks

7.2.1 Preamble Sampling Protocols

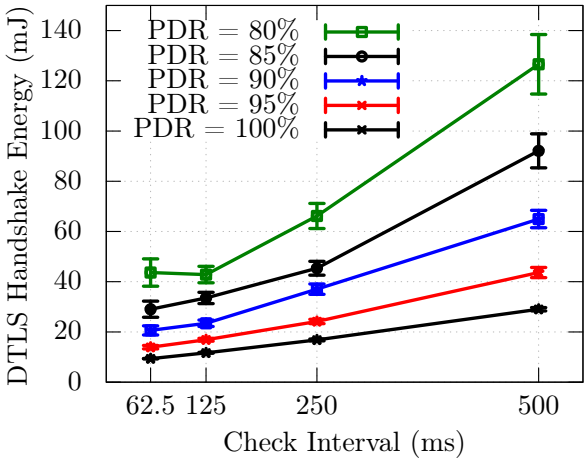
In this section, we use the preamble sampling protocol X-MAC [17] and its implementation in Contiki Operating System The period at which a node wakes up is called the "Check Interval" (`Check Interval (CI)`); the lower the `CI`, the more often nodes check the medium, and the higher their idle radio duty-cycle. Before sending a data packet, the transmitter repeatedly transmits a special control frame (called *strobe*) for at least the `CI`. For more details on preamble sampling techniques, the reader should refer to Chapter 3.



(a) DTLS handshake duration, single-hop.



(b) DTLS handshake duration, multi-hop (PDR=100%).



(c) Energy, single-hop.

Figure 7.2: Cost of a DTLS handshake in preamble sampling protocols.

We leverage tinyDTLS[1], an open-source DTLS implementation and its port to the Contiki operating system [151]. We use a pre-shared key cipher suite of DTLS with Advanced Encryption Standard (AES) operating in Counter Mode Encryption and Cipher Block Chaining Message Authentication Code (CCM) mode with 8-byte long authentication tags (`TLS_PSK_WITH_AES_128_CCM_8`). We evaluate the performance of this implementation by emulation, using the instruction-level MSP430 emulator MSPSim, and the Contiki Cooja simulator [113]. We emulate WiSMote, a popular constrained IoT platform featuring a 16-bit MSP430 micro-controller with 16 kB of RAM running at 12 MHz, and the IEEE 802.15.4-compliant CC2520 radio. The binary file used for emulation runs on WiSMote hardware.

In the Cooja simulator, we create a linear network of 2 to 5 nodes, depending on the required number of hops. The DTLS client (on one end of the linear network) repeatedly performs the handshake with the DTLS server (on the other end). There is no other traffic in the network. Similarly to Chapter 6, we estimate the energy consumption using Energest, a Contiki per-component profiling tool that measures the consumption of both the micro-controller and radio. We average the results over 1000 DTLS handshakes and present with a 95% confidence interval.

Overall results. Fig. 7.2 shows the measured average handshake duration and the energy consumption when DTLS runs on preamble sampling protocols, in the single/multi-hop case, and for different values of `CI` and link PDR[2]. The energy consumption in Fig. 7.2(c) is that of the DTLS client (running at 2.8 V), during the DTLS handshake. Although absolute energy values are specific to WiSMote, this platform is representative of hardware commonly deployed today, and the trends in Fig. 7.2(c) apply to all platforms. Overall, a DTLS handshake takes 1–50 s, with an energy consumption in 10–200 mJ range.

Impact of `CI`. At PDR=100%, the DTLS handshake duration and energy consumption grows linearly with `CI`. This is expected, as a larger `CI` reduces the rate at which nodes can exchange packets (hence a longer duration).

Impact of the number of hops. Similarly, separating DTLS server and client by additional hops increases the duration of the handshake. For PDR=100%, the increase is linear (some retransmissions still occur due to the hidden terminal problem); for PDR<100%, the increase is faster as a packet can be lost on each of the hops.

Impact of PDR. In any wireless environment, external interference and multi-path fading cause the PDR to be below 100%. When a DTLS message is lost, a timeout event occurs at the DTLS layer, which triggers retransmission of DTLS messages. X-MAC implicitly recovers from lost *strobes*, but does not detect failed receptions of data frames as there are no link-layer acknowledgements. This means that, when a DTLS packet is lost, the DTLS implementation waits for 2 s (the default tinyDTLS timeout value) before resending, causing a longer latency. Dropping

[1] `http://tinydtls.sourceforge.net/`

[2] Packet Delivery Ratio (PDR) of a link is the percentage of frames successfully received at the receiver node. We use the same PDR for all links in the emulated network and control its level through the Unit Disk Graph (UDG) radio model of Cooja.

the PDR from 100% to 90% roughly triples the handshake duration (Fig. 7.2(a)) and doubles the energy (Fig. 7.2(c)).

Energy Consumption. Fig. 7.2(c) shows how a DTLS handshake consumes more energy with a larger wakeup interval (longer sleep periods): increasing `CI` requires a transmitter to send a longer preamble. At PDR=100%, this increase is linear. To put this energy into perspective, a DTLS handshake cost of 29.05 mJ (`CI` =500 ms, PDR=100%) represents a consumption 5 orders of magnitude lower than the energy available in a pair of AA batteries. A single DTLS handshake has hence a negligible effect on the constrained node lifetime. The cost of completing a single DTLS handshake might be more prohibitive for energy harvesting nodes with small rechargeable batteries, For example, GreenNet [c_2] nodes have a 20 mAh battery, or 201.6 J at 2.8 V. In this context, a single 29.05 mJ handshake accounts for 0.0144% of the maximum available energy.

Packet overhead. Once the DTLS session has been established, DTLS with AES_CCM_8 cipher adds 29 bytes to each datagram (including an 8-byte nonce and 8-byte authentication tag), which represents 22.8% of the available link-layer payload space (127 bytes). For the details on byte overhead and possible optimizations, see Raza *et al.* [127].

7.2.2 Beacon-Enabled IEEE 802.15.4 Networks

Nodes in a beacon-enabled IEEE 802.15.4 network are organized as a cluster tree: some nodes are cluster heads (or coordinators), others are leaf nodes. Cluster heads periodically send beacon frames. A beacon frame indicates the start of a Contention Access Period (CAP), during which leaf nodes associated with the emitting cluster head communicate using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) medium access. After the CAP, and before the next beacon, all nodes switch their radio off. Because beacons are sent periodically, leaf nodes know when to wake up for the next CAP. The beacon interval (`BI`) and the duration of the CAP (`CAP`) are tunable, allowing a trade-off between the amount of data that can be exchanged, and the radio duty cycle. We do not use the Contention Free Period of beacon-enabled mode.

In our experiments, the DTLS client runs on a leaf node. In the single-hop case, the DTLS server runs on the cluster tree root, otherwise as a leaf node associated with the cluster tree root. We force a desired topology by tuning the parameters that specify the maximum number of associated cluster heads/leaf nodes such that the association requests are rejected until we obtain the desired topology. That is, we chain intermediate cluster head nodes between the DTLS client and the cluster tree root to obtain a linear network from 2 to 5 nodes, depending on the required number of hops. These cluster heads do not generate any application traffic; they only transmit periodic beacons and forward packets exchanged between DTLS server, associated with the cluster tree root, and the client.

We evaluate beacon-enabled mode on GreenNet nodes [c_2]. We port the tinyDTLS implementation to the GreenNet stack and modify it to use the AES hard-

ware acceleration block. We run tinyDTLS with exactly the same configuration as in Section 7.2.1 and estimate the energy consumption using Energest. We then derive energy by multiplying the values by the supply voltage and the current draw values from appropriate data sheets.

We obtain all the results in this section from measurements on GreenNet hardware. They are averaged over at least 500 handshakes and presented with a 95% confidence interval.

Overall results. Fig. 7.3 shows the measured average handshake duration and the energy consumption when DTLS runs on an IEEE 802.15.4 beacon-enabled network, in the single/multi-hop case, and for different values of `BI` and `CAP`.

Impact of `BI`. Results in the single-hop case (Fig. 7.3(a)) show how a short `BI` shortens the handshake as nodes get more frequent opportunities to transmit. Similarly, a large `CAP` gives nodes a long period to communicate; largest evaluated CAP of 123 ms yields shortest duration of the handshake. A larger `CAP` increases the throughput between two nodes, which means that a DTLS endpoint (client or server) can send its messages within the same CAP.

Impact of the number of hops. Fig. 7.3(b) presents the measured DTLS handshake duration when DTLS client and server are separated by multiple hops. As expected, the DTLS handshake duration increases almost linearly with the number of hops. For values of `BI` above 250 ms, the successful completion of a DTLS handshake between client and server multiple hops away requires the configuration of a large retransmission timeout value, even when there are no packets lost in the network. We observe handshake durations from 1.88 s to 16.6 s.

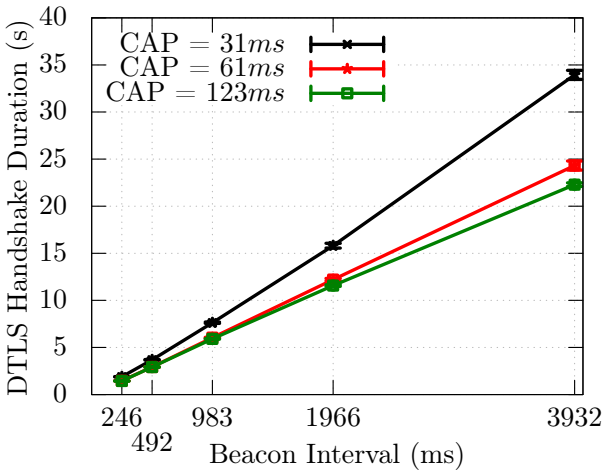
Energy Consumption. Fig. 7.3(c) shows the energy consumed by an GreenNet board running as a DTLS client during a DTLS handshake. The energy consumption only very slightly increases with `BI`, as the energy consumption of a transmitting node in beacon-enabled mode is not a function of the wakeup interval. Why the energy increases at all with `BI` is a consequence of the energy spent by the nodes when sitting idle.

7.2.3 IEEE 802.15.4e TSCH Networks

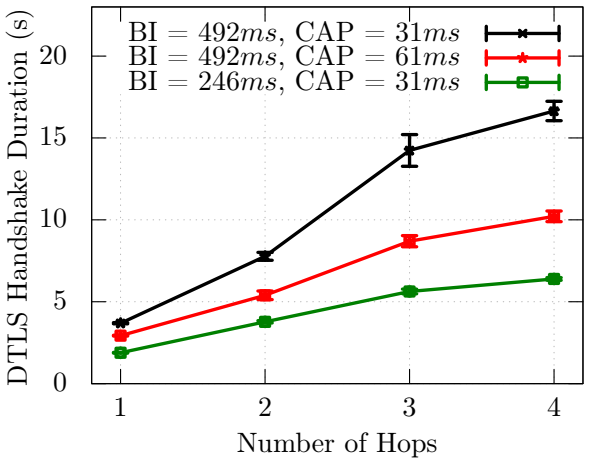
We derive the expected latency in a TSCH network analytically, and apply it to DTLS. For more details on TSCH mode, the reader should refer to Chapter 3. We assume that the centralized scheduler schedules a cell (a [timeslot, channel] pair) to only a single pair of nodes, thereby avoiding self-interference in the network.

Let C denote the number of cells scheduled in a slotframe between two nodes in a TSCH network and L the number of timeslots in a slotframe. We consider that cells are distributed in the TSCH schedule in a uniform fashion, i.e. the probability for a cell to be assigned to the appropriate timeslot is $1/L$. Consider a single-hop communication between two nodes; we are interested in finding the average latency D that includes the time a frame spent in a node queue before its transmission and reception at the destination node.

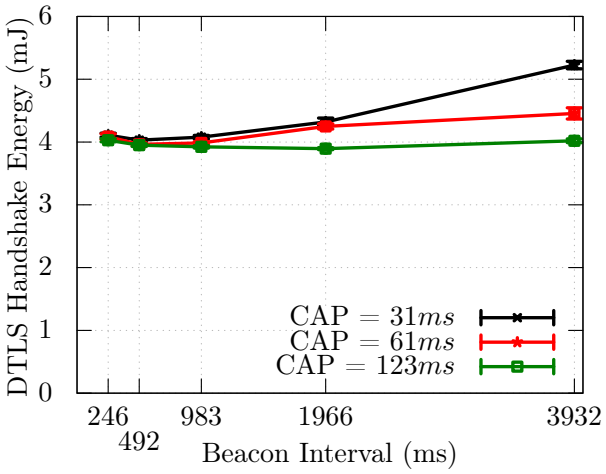
Let random variable T denote the instant in a slotframe when a frame has



(a) DTLS handshake duration, single-hop.



(b) DTLS handshake duration, multi-hop.



(c) Energy, single-hop.

Figure 7.3: Cost of a DTLS handshake in a beacon-enabled IEEE 802.15.4 network.

Table 7.1: Single-hop DTLS handshake duration in a TSCH network.

	C = 1	**C = 2**	**C = 3**
L = 101	5.15s	3.467s	2.625s
L = 1001	50.15s	33.467s	25.125s

been selected from a node queue for transmission. We consider T to be uniformly distributed over the slotframe length. Note that the instant T corresponds to a frame that is either self-originated, or received from a neighbor and to be forwarded. Let $X_0, X_1, \ldots, X_{C-1}$ denote random variables that correspond to the interval from instant T until the beginning of the corresponding cell slot. The average latency until the beginning of the frame transmission is the expectancy of the random variable $Y = min(X_0, X_1, ..., X_{C-1})$. Since the slotframe repeats in time, variables $X_0, X_1, \ldots, X_{C-1}$ are also uniformly distributed on $(0, L-1)$. Assuming $L >> C$, the average latency until the beginning of the frame transmission is $L/(C+1)$ timeslots. Eq. (7.1) expresses the single-hop latency (in timeslots), taking into account the duration of the timeslot during which the frame is transmitted, and the Packet Delivery Ratio P over the link.

$$D_{singlehop} = (1 + \frac{L}{C+1}) \cdot \frac{1}{P} \tag{7.1}$$

To extend Eq. (7.1) to the multi-hops case, we take into account the varying number of cells on each link. Considering a centralized schedule, the total latency over H intermediate hops is the sum of individual hop latencies:

$$D_{multihop} = \sum_{i=1}^{H} (1 + \frac{L}{C_i+1}) \cdot \frac{1}{P_i} \tag{7.2}$$

We use Eq. (7.1) and Eq. (7.2) to estimate the average duration of a DTLS handshake for typical TSCH values. We consider the same scenario as experimentally evaluated in Sections 7.2.1 and 7.2.2 (10 link-layer frames carrying DTLS messages) and a default timeslot duration of $10ms$. Tables 7.1 and 7.2 present the estimated DTLS handshake duration for typical values of L and C and ideal Packet Delivery Ratio.

We compared analytical results of TSCH with experimental results of *beacon-enabled* mode in order to find scenarios where they perform similarly. For a slotframe length of 101 timeslots, estimated handshake durations with 1, 2, and 3 dedicated cells in TSCH case, roughly correspond to *beacon-enabled* [`BI` = 983 ms, `CAP` = 61 ms], [`BI` = 492 ms, `CAP` = 31 ms], [`BI` = 492 ms, `CAP` = 61 ms] configurations, respectively.

Table 7.2: Multi-hop DTLS handshake duration in a TSCH network (C=1).

	H = 2	**H = 3**	**H = 4**
L = 101	10.3s	15.45s	20.6s
L = 1001	100.3s	150.45s	200.6s

7.3 The Impact of Memory Constraints

We have so far focused on the communication aspects of DTLS in duty-cycled networks. As the role of a DTLS server is often assumed by a constrained device, this section focuses on the effect of memory limitations on DTLS session management. RFC 7228 [15] defines three classes of constrained devices: Class 0 ($\ll$ 10 kB RAM, $\ll$ 100 kB flash), Class 1 ($\sim$ 10 kB RAM, $\sim$ 100 kB flash), Class 2 ($\sim$ 50 kB RAM, $\sim$ 250 kB flash). According to this classification, WiSMote platform is a Class 1 device, while GreenNet falls in-between Classes 1 and 2.

Because of the way memory is allocated with embedded processors, a typical implementation statically allocates a number of DTLS "session slots", limiting the number of sessions simultaneously open. The memory footprint for a single DTLS session depends on the cipher suite and key length. The session state includes the IPv6 address and a port number of the communicating peer, its role (i.e. client or server), DTLS engine state, master secret, derived keys and other implementation specific variables. As an example, the tinyDTLS implementation uses $\sim$ 400 B of RAM per pre-shared key session, depending on the data type sizes used by different compilers, memory alignment and hardware architecture. That said, the operating system and the networking stack account for most of the available memory. For instance, with 16 kB of RAM available on WiSMote nodes, we could only fit 3 DTLS session slots together with the full ContikiOS and the IPv6 networking stack including tinyDTLS, and a simple application for evaluation purposes.

We therefore want to determine the probability P_B that a DTLS client attempts to establish a session with a DTLS server where all DTLS session slots are already occupied. We call R the number of DTLS session slots available at a server. Let N denote the number of clients interested in establishing a session with the given server. We model the individual client rate with the exponential distribution of parameter λ, and the duration of each established session with parameter μ. Generated traffic in Erlang by each client is then $\rho = \lambda/\mu$. Under these assumptions, the blocking probability P_B of a DTLS server is simply the blocking probability of a $M/M/R/R/N$ queue, i.e. the Engset loss formula.

For instance, if we consider $R = 3$ (the case observed with WiSMote), $N = 5$ DTLS clients with $\rho = 0.5$, the blocking probability of a request is $\sim$ 17%. A DTLS server may implement different strategies for handling such requests. It may discard them or decide to close one of the open sessions in order to accommodate the newly arriving one. In the latter case, performance depends on the session closure policy, i.e. the appropriate "cache" replacement algorithm.

7.4 Conclusion

The results of our evaluation reported in this chapter demonstrate surprisingly poor performance of DTLS in radio duty-cycled networks. Experiments with preamble sampling protocols show that the total duration of a DTLS handshake can be more than 50 s, depending on the Check Interval. In the case of the IEEE 802.15.4 beacon-enabled mode, we measure durations up to 35 s with the largest used Beacon Interval of ~ 4 s. Handshake duration increases linearly with the number of hops for both preamble-sampling and IEEE 802.15.4 beacon-enabled networks. We also derived the analytic expression for latency in TSCH networks and applied it to estimate the duration of DTLS handshake. For instance, in a typical TSCH network with 101 timeslots per slotframe, where DTLS server and client are radio neighbors and have a single dedicated cell for communication, it takes 5.5 s on the average to complete the handshake. This value decreases to 2.6 s for 3 dedicated cells.

Our results show that using DTLS is acceptable for applications for which a DTLS handshake is performed a limited number of times during the constrained node lifetime. For scenarios that require multiple DTLS clients per DTLS server with constrained resources, we study the blocking probability and show that it corresponds to the Engset loss formula. Applications expecting a large number of clients per DTLS server should cautiously weight the benefit of its use. Apart from these performance issues, DTLS is inherently incompatible with any sort of group communication and application-level, untrusted intermediaries. We discuss this in more details in Chapter 8 and consequently propose a system-level solution.

Chapter 8

OSCAR: Object Security Architecture for the Internet of Things

Although constrained nodes of Internet of Things (IoT) may benefit from the existing Internet Protocol (IP) security protocols, their core design assumptions build upon the connection-oriented security model that poorly fits IoT requirements. Research efforts towards the secure IoT have mostly concerned designing lightweight variants of security protocols and porting them to constrained nodes [50, 127, 125, 128]. However, they do not pervade sufficiently, which led to a situation in which the recently standardized Constrained Application Protocol (CoAP) [144] fully supports the application requirements, but security does not keep up.

Smart devices, due to their severe energy and memory constraints, heavily rely on group communication, asynchronous traffic, and caching. Supporting a variety of existing security protocols/mechanisms to specifically target these requirements is practically impossible due to memory limitations. Internet Engineering Task Force (IETF) has thus taken a position [144] to reuse Datagram Transport Layer Security (DTLS), the all-round point-to-point security protocol, to secure the communication channel between a constrained device running CoAP, in further text denoted as constrained CoAP node, and a client.

Apart from the performance issues discussed in Chapter 7, incompatibility with caching and multicast traffic, the DTLS approach has an important impact on scalability: Memory limitations of constrained nodes restrict the number of DTLS sessions.

In IoT scenarios such as Smart Cities in which a large number of clients may communicate with constrained CoAP nodes, the limitations lead to a considerable load that translates to an increased energy consumption and a shortened lifetime.

We follow the Representational State Transfer (REST) architecture model [41] to address this problem from a networking perspective and to remove the notion of state between server and client [c_4] [c_1]. We achieve this goal by leveraging the concept of *object security* that concerns data instead of communication end-points. In the proposed Object Security Architecture for the Internet of Things (OSCAR), we offload some expensive operations from constrained CoAP nodes to more powerful servers. Initially, constrained CoAP nodes publish their certificates to Authorization Servers and clients contact them to obtain Access Secrets that enable clients to request resources from constrained CoAP nodes. They reply with the requested

resources that are signed and encrypted. The scheme couples the object security principle with the capability-based access control and provides communication confidentiality and protection against replay attacks. We use secure channels established by (D)TLS for distribution of certificates and Access Secret.

The main contributions of this chapter are the following:

- A new scalable security architecture for IoT that jointly provides end-to-end security and access control, decouples confidentiality and authenticity trust domains, and intrinsically supports multicast, asynchronous traffic, and caching,
- An evaluation of the architecture in a constrained Machine to Machine (M2M) scenario for two hardware platforms and Medium Access Control (MAC)/Radio Duty-Cycling (RDC) protocols, on a real testbed and on the instruction level emulator of Cooja, demonstrating performance benefits with an increasing number of clients.

The chapter is organized as follows. We discuss the current Internet security model and the requirements of IoT applications in Section 8.1. We provide a detailed description of the proposed architecture in Section 8.2. In Section 8.3, we elaborate on how traditional Cloud services can integrate our architecture. We analyze and discuss security considerations in Sections 8.4 and 8.5, and present evaluation results in Section 8.6. We summarize in Section 8.7 main ideas that were interesting enough to be discussed and considered as potential solutions within standardization bodies. In that context, Section 8.8 briefs on the typical authorization flows and explains why concepts introduced by our architecture can be interesting from the authorization point of view. We conclude in Section 8.9.

8.1 Internet Security Model and IoT Requirements

The Internet relies on the communication model involving end-points. Security design followed by placing the trust on end-points and securing the communication channel. With evolving applications, the Internet has become a content distribution network leveraging the legacy client-server architecture. This paradigm has led to substantial research efforts on future Internet architectures, such as information centric networks, like DONA [80] and Content-Centric Networking [64]. Our work leverages their contributions and applies the general concepts with the goal of providing a robust, but flexible security approach to IoT and its traffic requirements.

As discussed by Smetters and Jacobson [146], the host oriented paradigm has a direct consequence on trust – its transitivity: Once a logical connection between the hosts is closed, the trust in the information is gone. The model serves very well typical e-commerce, e-banking, or IP telephony applications, because the trust in the information is implicitly dependent on the trust of the communicating entities *during the connection time*. However, considerable issues arise when the notion of a connection disappears. As stressed by Modadugu and Rescorla [107], Domain Name

System (DNS) is purposely secured with the application level Domain Name System Security Extensions (DNSSEC) and not with a connection-oriented protocol such as DTLS. Content oriented security schemes such as Secure/Multipurpose Internet Mail Extensions (S/MIME) or Pretty Good Privacy (PGP) secure electronic mails that pass multiple application level gateways without a clear connection between end-points. IoT applications behave similarly:

- *Application traffic is asynchronous.* Constrained CoAP nodes (event detectors, monitoring sensors, smart meters) notify their clients (subscribers) of measured values or physical state changes as they happen. Clients send commands to actuating devices asynchronously in reaction to the changes in the environment. DNS traffic is a good parallel as asynchronous human actions trigger name resolution.

- *Caching is a must.* Severe energy constraints of sensor nodes lead to long sleep periods with less than 1% of duty cycles. In this case, caching sensed data at untrusted intermediaries appears as an important means for keeping applications running independently. Electronic mails face a similar problem as they may go through untrusted servers until delivery.

- *Group communication is frequent.* In common IoT applications, clients may want to send messages to a subset of sensor/actuator nodes to perform an action, for example to turn off all lights on n^{th} floor or to update the firmware. IPv6 multicast and User Datagram Protocol (UDP) provide support for this type of traffic bearing no connection state between end-points.

Typical Web applications involve a single server and multiple clients [41]. As a consequence, the server side application may control access after client authentication. IoT reverses this paradigm by having many nodes serving as servers and possibly many clients taking part in the same application. More importantly, CoAP nodes may need to reduce their functionalities due to resource constraints. Subsequently, access control becomes a distributed problem, especially when taking into account the recent efforts for decoupling the sensor network infrastructure from applications [89, 40]. Furthermore, new applications have emerged that use local databases to store parts of collected data. For example, in Antelope [166] each constrained node in a network runs a database management system.

For these reasons, the connection-oriented security model does not fit well the actual IoT needs. Connection time tweaking and keep-alive messages could probably squeeze in connection-oriented security protocols, such as DTLS or IPsec, and work around the asynchronous traffic requirement. Aside the overhead and performance issues, this approach would still provide us only with the communication channel security. To support caching, we would need to trust the intermediate nodes/proxies to store the data in clear. To support group communications, we would need to open separate secure connections among group members and/or add additional protocols

on top of them, which effectively provides redundant security services necessary for use cases. Such a solution is not a long term approach.

Nevertheless, we need to keep the existing connection-oriented security protocols in the overall IoT picture. In fact, our OSCAR approach relies on secure and authenticated channels established by DTLS for certificate and Access Secret distribution. We couple the concepts of connection-oriented security with those of content-centric networking [64].

8.2 OSCAR

In this section, we introduce the OSCAR architecture and present its main advantages.

8.2.1 Technological Trends and Design Goals

The following trends have guided our design:

- Constraints on energy are almost constant. Without a breakthrough in battery chemical engineering, the available energy is expected to remain the main constraint for IoT devices.
- Available memory for embedded devices slowly increases. However, due to the economical and energy costs caused by leakage, we expect that memory will remain limited and a determining factor for the unit price.
- Processing capabilities constantly increase even for ultra low-power micro controllers. Thus, we do not see the processing power as a limiting constraint in the future.

Apart from the sleep mode leakage, radio communications are the main energy consumer. Thus, our primary design goal is to minimize the number of frames/packets to transmit or receive for security purposes. We achieve this goal by leveraging the benefits of public-key cryptography, sparse traffic patterns within constrained node networks, and messages of limited size – we trade the radio usage for a higher computation load.

8.2.2 Producer-Consumer Model

We consider IoT, its sensors and actuators, as an interface to the physical world. Decision takers (human users, intelligence centers, or constrained actuating devices themselves) base their reasoning and actions on data coming from the sensed physical phenomena. The relation between the decisions or actions and the sensed phenomena is *many to many* – a single measurement data may affect multiple decisions and a single decision may require many different measurements.

Consider for instance a traffic control application in a Smart City. A traffic light management subsystem may use the current traffic intensity and pollution readings

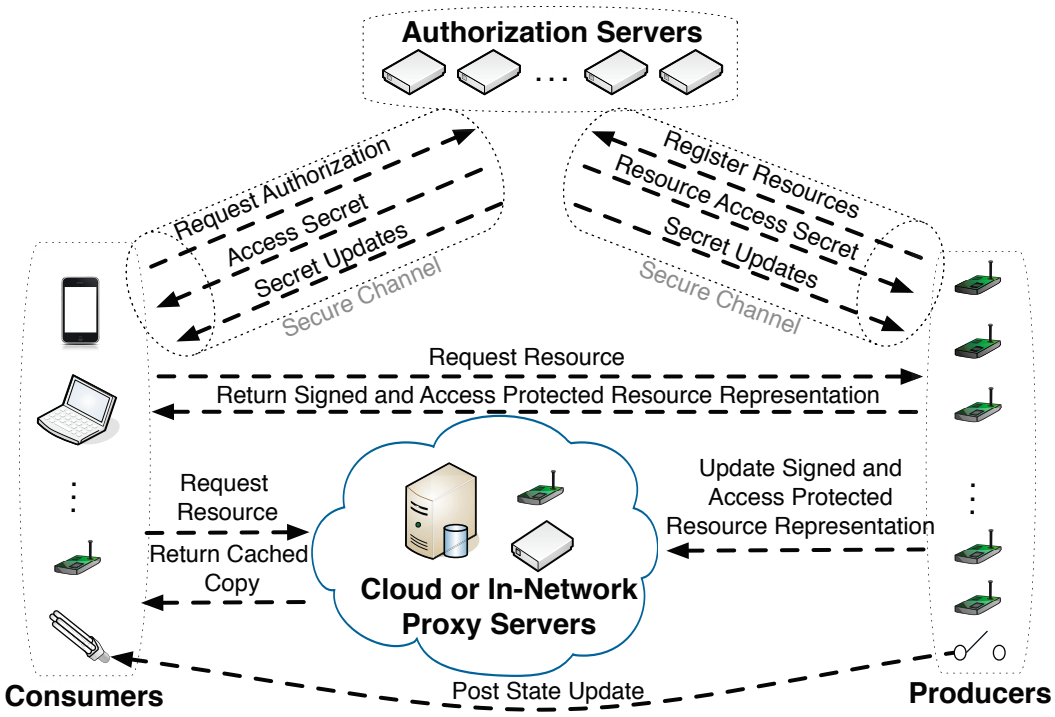


Figure 8.1: OSCAR, an object-based producer-consumer security architecture.

from all over the city as input data for control decisions. At the same time, local readings may influence decisions made on luminosity of nearby street lights.

The producer-consumer model represents well the relations within IoT. *Producers* (smart meters, traditional sensors, motion detectors, etc.) feed *Consumers* with the required information. Consumers (actuating devices, collection centers, human users) gather up the information and may further generate actions. Cloud Computing and the recent work on data access control [69] take a similar view, however, Producers in the IoT case are not access control decision makers for the content they generate, which is rather a policy of the network owner.

8.2.3 OSCAR Security Architecture

Fig. 8.1 presents the OSCAR security architecture. Its main components are the following:

- Producers: Constrained CoAP nodes that provide data in the form of signed and encrypted resource representations (temperature, humidity, CO_2, etc.).
- Consumers: CoAP clients that request resource representations from Producers.
- Authorization Servers: They are trusted entities that store certificates of Producers, receive registrations of Producers for generated resources, and provide

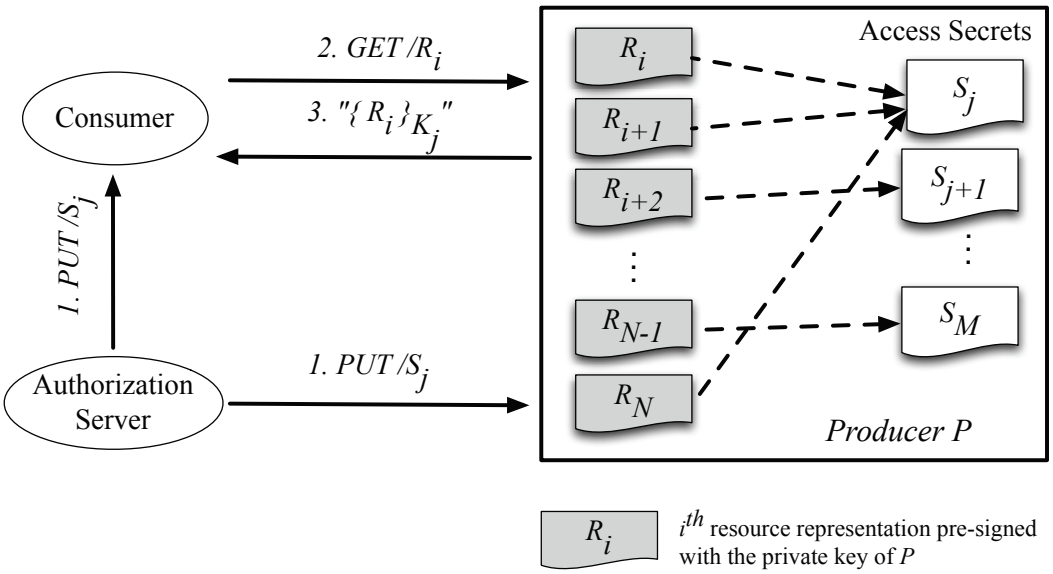


Figure 8.2: Principles of accessing resource representations on a Producer.

Access Secrets protecting Producer resource representations. When a Consumer requests access to a Producer resource, an Authorization Server returns the Access Secret that allows the reception of the Producer resource representation.

- Proxy Servers: Provide a caching service between Producers and Consumers to hide unavailable Producers (when for instance constrained nodes go to sleep to save energy). They present the same interface to Consumers as Producers.

We assume that Producers and Consumers have valid certificates issued by a Certification Authority and they have root certificates. A Producer uses its private key to sign resource representations. An Access Secret is a token generated by an Authorization Server from which a Producer derives a symmetric encryption key to encrypt a resource representation. Access Secrets have their corresponding public identifiers and can be shared among multiple Producers to protect a common resource. For example, all temperature readings in a building may be protected using the same Access Secret. Producers/Consumers and Authorization Servers use a secure DTLS session to exchange cryptographic materials (Access Secrets and certificates).

Fig. 8.2 presents the principles of accessing resource representations on a constrained CoAP node with the role of Producer P. P manages a set of resources $R_i, 1 \leq i \leq N$ and a set of Access Secrets $S_j, 1 \leq i \leq M$ obtained from an Authorization Server. Access Secret S_j defines a group of access rights allowing for different authorization levels. The relation between j^{th} Access Secret S_j and i^{th} resource R_i depends on authorization policies. As a consequence, this introduces a memory overhead as it defines the total number of Access Secrets that P needs to

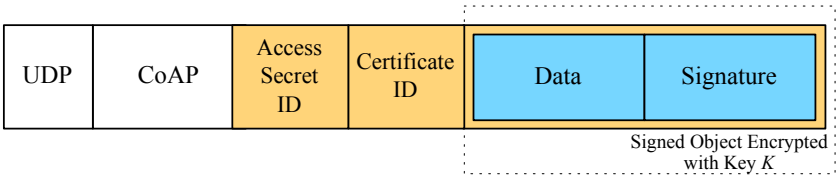


Figure 8.3: Structure of the signed and encrypted resource in a CoAP message.

locally store.

$\{X\}_{K_j}$ denotes symmetric encryption of X with key K_j derived from Access Secret S_j:

$$K_j = f(S_j, MessageID, CertificateID), \tag{8.1}$$

where $f()$ is a generic pseudo-random function, $MessageID$ is the message identifier in the CoAP header, and $CertificateID$ is the node unique identifier contained in its certificate.

Consumers/Producers and Authorization Servers manage Access Secrets as resources so an Authorization Server can use the idempotent PUT method of CoAP to create or update them. Once a Consumer obtains Access Secret S_j from an Authorization Server, it can then invoke the GET method of CoAP on resource R_i. The Producer returns the resource signed with the Producer private key and encrypted with key K_j. Fig. 8.3 presents the structure of the signed and encrypted resource. Only the Consumer that has Access Secret S_j can decrypt the resource. Note that the Producer pre-signs resources as soon as they become available.

We bind the certificate of a Producer or a Consumer with device firmware and thus include the list of supported ciphers in the certificate itself. Producers and Consumers can then learn about their supported ciphers from certificates distributed by the Authorization Server and avoid the cost of cipher negotiation. To support this way of operation, we require an additional Accept option in the CoAP header to carry the cipher chosen by the party initiating the request or simply new content types corresponding to different cipher suites protecting object security payload.

8.2.4 Cryptographic Overhead

OSCAR ensures authenticity and integrity of resources by leveraging digital signatures, which may seem surprising as we target constrained devices with limited computational resources. However, the use of public-key operations at the level of semantic content allows decoupling the server-side cryptographic overhead from network communication: Producers can update their resource representations whenever the semantic changes, when it suits their schedule (take for example energy-harvested devices) and more importantly, while the radio transceiver is turned off. The burden of digital signature verification is then put on Consumers that have sufficient computational resources. In scenarios where Producers need to verify digital signatures, like for instance with actuating devices, it is possible to cache already

verified signed objects, *i.e.* actuating commands, and reduce the cost of signature verification to a simple cache lookup.

8.2.5 Packet Overhead

Fig. 8.3 illustrates that apart from the cipher specific overhead, OSCAR requires two additional fields with each transmitted packet: 1) Access Secret ID, an identifier of the Access Secret used for the encryption key derivation; 2) CertificateID, an identifier of the node originating the signed object, that is related to the appropriate certificate. On the reception side, Access Secret ID is used to index the corresponding Access Secret, and if not locally available to request it from the Authorization Server. Similarly, CertificateID is first included in the key derivation procedure, and then used to look up the certificate needed for signature verification. In case the certificate is not locally available, it can be requested from the Authorization Server. The size of the two fields may vary according to the deployment requirements and if cross-domain communication is desirable. In our prototype implementation, we use one byte long domain unique identifiers. A major part of the packet overhead, however, comes from the digital signature. For example, in the case of Elliptic Curve Cryptography (ECC) and secp160r1 curve, the length of the signature is 40 bytes, while in the case of a larger secp192r1 curve, it spans 48 bytes. We communicate the OSCAR payload type together with the used ciphers within CoAP Content-Format option, adding 2 bytes for signaling with respect to the minimal CoAP header size.

8.2.6 Implementation and Standardization Requirements

Components required for the implementation of OSCAR are already available in open source form, for instance one can fully leverage the already available implementations of cipher suites used by DTLS. In effect, the only building block of OSCAR that needs to complement a DTLS implementation is an object security parsing library, introducing small memory overhead. For example, our prototype implementation with custom object security and certificate formats, tailored for constrained devices, introduced additional 2156 bytes of ROM overhead for the parsing library and up to 500 bytes of RAM in order to store the minimal necessary cryptographic material – ECC certificate of a device, the corresponding private key and the root trust certificate. On top of this, depending on the application requirements, varying number of Access Secrets and other certificates may be cached to improve the performance and avoid communication with the Authorization Server. Operating at the application layer OSCAR does not require any changes in the operating system kernel.

8.2.7 Examples

In this section, we give two examples of GET and PUT operations. Algorithm 7 shows the steps for a Consumer (C) requesting a temperature reading by means of a

GET request to a Producer (P). Communication in Steps 1-6 in both examples uses a secure authenticated channel established by DTLS. Notice that the Consumer needs to go through Steps 1-6 only at the beginning and the Authorization Server (AS) can later update the necessary Access Secret according to the key management scheme and authorization policies as illustrated in Fig. 8.1.

Algorithm 7 Example of a GET request for `example.net/temperature`

1: **P** → **AS**: `example.net` has resource "temperature"
2: **AS** → **P**: S_1 is the Access Secret for resource "temperature"
3: **C** → **AS**: Request authorization for "GET URI-Host:`example.net` URI-Path: `temperature`"
4: **if AS** grants authorization to **C then**
5: **AS** → **C** : (Certificate of P)
6: **AS** → **C** : S_1, it allows access to "`temperature`" at "`example.net`"
7: **C** → **P**: CoAP GET to `example.net /temperature`
8: **P** → **C**: " 25.5 °C ", signed with the private key of P and encrypted with a key derived from S_1
9: **C** decrypts `temperature` and verifies the signature

Algorithm 8 Example of a PUT/POST request for `example.net/frontDoor`

1: **P** → **AS**: `example.net` has resource "`frontDoor`"
2: **AS** → **P**: S_2 is the Access Secret for resource "`frontDoor`"
3: **C** → **AS**: Request authorization for "PUT URI-Host:`example.net` URI-Path:`frontDoor` Payload:`open`"
4: **if AS** grants authorization to **C then**
5: **AS** → **C**: S_2, it allows access to "`frontDoor`" at "`example.net`"
6: **C** → **AS or P**: (Certificate of C)
7: **C** → **P**: CoAP PUT to `example.net frontDoor` Payload: "`open`", payload signed by C and encrypted with a key derived from S_2
8: **if P** decrypts the payload with the key derived from S_2 **then**
9: **if P** verifies the signature of C **then**
10: **P** sets `frontDoor` to "`open`"

The procedure for PUT/POST requests is very similar to GET (cf. Algorithm 8). P needs to decrypt and verify the payload of the request to decide if it is going to grant it. P can learn the necessary certificate either directly from C or request it from the Authorization Server. In the case of DELETE requests, a Consumer would simply need to attach in the payload an encrypted and signed resource encapsulating a pre-defined token. Similarly to the first example, P can locally cache different signed resources to avoid signature verification for each request. As a consequence, the impact on latency can be reduced to network communication delays, with the processing overhead comparable to the case without security. This yields the effect of OSCAR on user-triggered actions minimal, while fully leveraging the benefits of

digital signatures and source authentication.

8.3 Integrating the Internet of Things with the Cloud

OSCAR enables secure integration of data generated within an IoT domain with various Cloud services by leveraging the capability-based access control. However, binding the encryption key with the underlying CoAP header prevents storing encrypted resources in the Cloud, as the key cannot be derived once the information from the CoAP header is lost. To support access to IoT data in the Cloud independent of CoAP, Authorization Servers can communicate two Access Secrets for a single resource to Producers – Access Secret S_L for protection on the communication path, and Access Secret S_H for protection of the content. The idea is to generate by a Producer a double-encrypted payload that encapsulates the signed resource. The Producer can first encrypt offline the signed resource with Access Secret S_H and then, encrypt it the second time with S_L for the transmission over the network. Therefore, Access Secret S_H is used for the actual access to the data in the Cloud, while the purpose of Access Secret S_L is to protect from communication-related attacks, while the data is traversing the network (cf. Fig. 8.4).

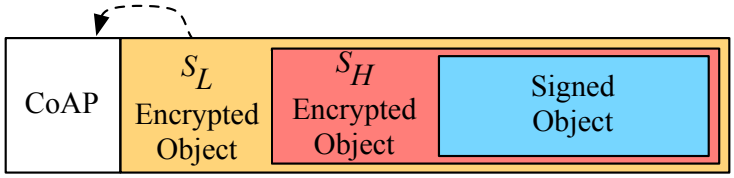


Figure 8.4: Double-encrypted resource traversing the network to support access-protected storage in the Cloud. S_L and S_H denote two independent Access Secrets. The arrow represents that the encryption key is derived as a function of the underlying CoAP header fields.

We assume the presence of a Cloud host that runs CoAP, has S_L, and has subscribed to receive the updates of different resources from Producers (cf. Fig. 8.1). Its role is to decrypt the first encrypted payload bound to the CoAP header and to store its payload – an access-protected and signed resource. Note that Authorization Servers provide access to Consumers of resources by sharing with them Access Secret S_H.

8.4 Replay Attack Analysis

Protecting the communication between a Producer and a Consumer against replay attacks requires maintaining some state between them, which contradicts our goal of providing a stateless approach to security. The way of deriving the encryption key from the Access Secret and a $MessageID$ allows the Consumer to detect a resource replayed by an external attacker. However, this approach is vulnerable to

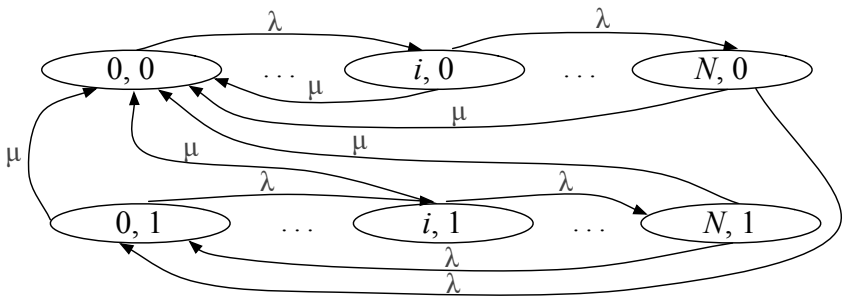


Figure 8.5: Continuous-time Markov chain model for the probability analysis of replay attacks with state space $Z = \{i, j\}$, where $i \in [0, N]$ is the number of messages sent since the last Access Secret update and $j \in \{0, 1\}$ is the possibility of the replay attack. λ is the rate of outgoing messages and μ is the rate of Access Secret updates initiated by Authorization Servers.

replay attacks once the Producer or a Consumer looses the $MessageID$ context or $MessageID$ wraps up. To overcome this problem, we rely on updates of the Access Secret provided by the key management scheme. Given that the Access Secret is shared among the members of the group, possibly within the same local network, its update may trigger a large communication overhead for the constrained network. For this reason, the rate of Access Secret updates should be chosen in order to reflect the tradeoff between security and network performance. Furthermore, in some use cases like overseas container monitoring [139], Internet connection may not be available at all times, rendering the remote Authorization Server unable to enforce authorization decisions in real time. Therefore, in this section we consider random Access Secret updates and obtain insights how the performance of the system can be optimized depending on different traffic patterns.

We analyze the vulnerability to the replay attack by an external attacker under the assumption that $MessageID$ handling of CoAP is implemented such that it can provide long-lived duplicate detection. In that sense, we assume that Producers and Consumers monotonically increase local $MessageID$ variables and keep track of their communicating peers. We model the evolution of $MessageID$, local to the sender of a given CoAP message containing a resource payload, with a continuous-time Markov chain illustrated in Fig. 8.5. N represents the maximum number of $MessageID$ increments before the variable wraps up. Parameter λ is the rate of outgoing messages and parameter μ is the rate of Access Secret updates initiated by the Authorization Server. In CoAP specification [144], $MessageID$ uses 16 bits allowing for the maximum of $2^{16} - 1$ uniquely identified messages.

Once the $MessageID$ wraps up, a network adversary can replay old messages with a $MessageID$ greater than the current one. Since the encryption key of the message payload depends on $MessageID$, such an injection would pass unnoticed at the Consumer, and it would accept an old message as a fresh one. Updating Access Secrets over time can protect the nodes from such an attack. We need to keep the probability of the attack as low as possible by using different Access Secrets, which will help us to parametrize the frequency of Access Secret updates as a function of

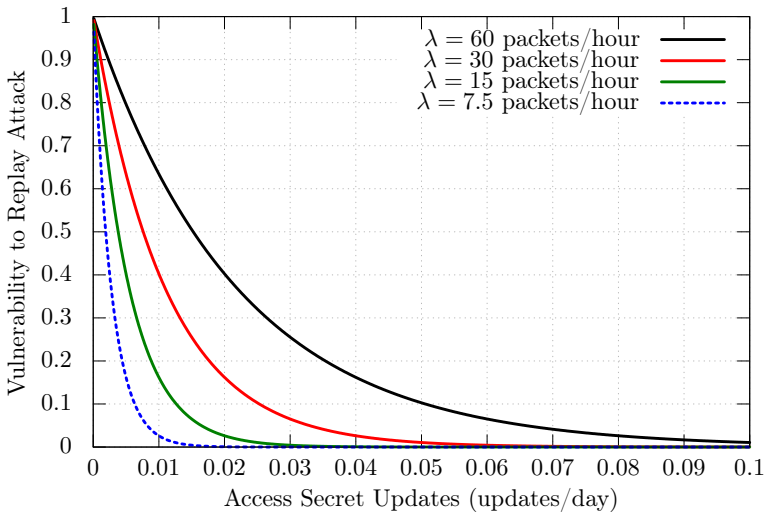


Figure 8.6: Vulnerability to replay attacks for a varying Access Secret update rate.

the expected λ.

As we can notice in Fig. 8.5, a node enters the states at which replay attacks are possible, if there was no update of the Access Secret by the time $MessageID$ wraps up. At any instant, the update of the Access Secret effectively resets the Markov chain to the initial state $\{0, 0\}$. We are interested in finding the sum of stationary probabilities $\pi_{i,j}$ of the states at which the replay attack is possible:

$$P_{replay} = \sum_{i=0}^{N} \pi_{i,1} \tag{8.2}$$

We can observe that:

$$\pi_{i,1} = \pi_{0,1}(\frac{\lambda}{\lambda+\mu})^i, i \in [1, N]; \pi_{0,1} = \pi_{0,0}\frac{1}{(\frac{\lambda+\mu}{\lambda})^{N+1} - 1}; \pi_{0,0} = \frac{\mu}{\lambda+\mu} \tag{8.3}$$

From Eqs. 8.2 and 8.3, with $\rho = \lambda/\mu$, it follows that:

$$P_{replay} = \pi_{0,0}\frac{1}{(\frac{\lambda+\mu}{\lambda})^{N+1} - 1}\sum_{i=0}^{N}(\frac{\lambda}{\lambda+\mu})^i = \frac{1}{(1+\frac{1}{\rho})^{N+1}} \tag{8.4}$$

Since the encryption key is bound to the Producer unique identifier (cf. Eq. 8.1), probability P_{replay} is independent of the number of nodes sharing the same Access Secret, which is particularly important for the scalability of OSCAR. We plot this probability in Fig. 8.6 for $N = 2^{16} - 1$ and a varying Access Secret update rate.

8.5 Security Considerations

Denial of Service: OSCAR takes a non-traditional approach to fight Denial of Service. It builds upon the assumption that typical IoT resource representations are small in size (individual measurements of physical quantities, actuator state changes) and directly responds to requests with access-protected resource representations. Moreover, it does not keep any state between communicating entities, which we find particularly important to fight memory exhaustion attacks. Note also that since digital signing operations are done offline, the intensity of incoming traffic is not correlated with asymmetric cryptographic overhead. Clearly, in the case of large resource representations such an approach is a security concern. Producers could limit the response rate locally and define a resource size threshold above which a Consumer would need to include an encrypted object in the payload of a request, proving the possession of the appropriate Access Secret.

Confidentiality: As nodes use resource encryption keys derived from Access Secrets, OSCAR provides confidentiality within the resource access right group. The actual security properties depend on the algorithm used for encryption. Note that if an adversary is able to compromise Authorization Servers, it may only eavesdrop – E2E integrity and authentication properties are preserved.

Replay Protection: Another concern related to the replay attack is a malicious adversary within the resource access right group. Such an "insider" can inject old resource representations making other members of the group believe they are fresh (if within the content itself, there is no means allowing the detection of an old reading/command, *i.e.* a timestamp). The protection of nodes from such adversary would require the use of time within signed objects for replay protection or a more complex Access Secret management scheme/protocol. For instance, one could use the recently proposed one-to-many scheme by Szalachowski and Perrig [152] that achieves asymmetry by using Berkovits' protocol [13] and Shamir's secret sharing scheme [143]. The threat model assumed by Szalachowski and Perrig allows the attacker to be within the "privileged" group, which in our case corresponds to nodes sharing an Access Secret.

Integrity Protection of the CoAP Header: Since OSCAR does not provide integrity protection of the CoAP header, a network adversary can launch attacks by altering its fields. For instance, an adversary could replace Uniform Resource Identifier (URI) of a PUT/POST/DELETE request with another one in case the resources are protected with the same Access Secret. To protect nodes from such attacks, it would suffice to include different fields of the CoAP header in the key derivation procedure of Eq. 8.1 (URI path, method code, token, options, etc.).

Node Compromise: Similarly to the most Internet security protocols, OSCAR in its design does not assume adversary's capability to physically compromise end-

points (producer, consumer) and the extraction of sensitive cryptographic material (private keys, Access Secrets). However, given that in many IoT deployments nodes may be physically accessible by third parties the risk should not be neglected. Note, however, that the compromise of intermediary proxies, from the security point of view, is oblivious to other participants in the architecture, as proxies are not required to decrypt secured objects and therefore do not keep Access Secrets in their memory. On the other hand, in case producers or consumers can be tampered with, OSCAR may be extended with an independent scheme for detection of compromised nodes which should trigger the update of affected Access Secrets in the network. An adversary in possession of an appropriate access secret and the private key is able to inject false data and more importantly, to control actuators within the deployment. Therefore, to mitigate this risk vendors are encouraged to provide tamper-resistant components with their IoT devices.

8.6 Performance Evaluation

We have implemented an OSCAR library with custom object security format tailored for constrained devices under the Contiki operating system. The library builds upon the open source implementation of ECC cryptographic primitives – TinyECC [95]. We use Advanced Encryption Standard (AES) in Extension of Counter Mode Encryption and Cipher Block Chaining Message Authentication Code (CCM*) mode [37] for symmetric encryption. The library supports creation, parsing, and verification of "encrypted" and "signed" object types. A certificate is then just a particular type of a "signed object" with a pre-defined format, providing the binding of a public key with device identity. We have coupled the object security library with Erbium CoAP, a default CoAP implementation for Contiki (version 07) to add cipher suite negotiation capabilities.

We have evaluated three important aspects of OSCAR: 1) The computation overhead of Elliptic Curve Digital Signature Algorithm (ECDSA) on constrained nodes, 2) Scalability in M2M communication scenarios, 3) The impact of radio duty cycling mechanisms on the performance. We have performed our experiments on two hardware platforms that represent the characteristics of two generations of IoT devices:

- WiSMote board [179] based on 16-bit MSP430 (series 5) Microcontroller Unit (MCU) with 16 KB of RAM and an 802.15.4-compatible CC2520 radio transceiver. We obtain the WiSMote related results using the instruction level MSP430 emulator MSPSim and the Contiki simulator Cooja [113]. We have confronted the results of the ECDSA overhead from the emulator in Cooja with those obtained on the real WiSMote hardware and we have measured a maximum error of 2.67%.

- GreenNet board [c_2], based on an ultra low power 32-bit ARM Cortex-M3 MCU (STM32L) with 32 KB of RAM and an 802.15.4 radio transceiver. Mote

details on GreenNet boards are available in Chapter 2. We have obtained the reported results on an operational hardware platform.

To eliminate the effect of a variable CPU frequency on results, we have configured both platforms at 21.3 MHz. MSP430 series 5 may be configured up to 24 MHz, while the STM32L can be clocked up to 32 MHz. Both computation time (inversely proportional) and MCU energy consumption (directly proportional) are linearly dependent on frequency.

We estimate energy consumption using Energest, the Contiki per-component profiling tool.

8.6.1 ECDSA Computation Overhead

In Chapter 4, we presented the evaluation of ECDSA overhead for GreenNet and WiSMote boards. See Figs. 4.2(a) and 4.2(b). The computation overhead ranging from 0.3 to 0.9 seconds for GreenNet and from 1.18 to 3.63 seconds for WiSMote may seem a huge price to pay. In fact, Hummen *et al.* argued that for this reason, we should minimize the number of public-key operations during the security handshake [61]. OSCAR, however, compensates for this overhead by removing the radio energy cost of the security handshake with every Consumer.

These figures for two generations of IoT devices strongly support our initial design assumption on processing capabilities (cf. Section 8.2.1). Whatsoever, we expect that further advancements in the chip manufacturing technology will additionally reduce the energy and computation costs for low power MCUs.

8.6.2 Scalability

Our goal in this section is to determine if OSCAR and the heavy use of ECC public-key primitives outperform a connection-oriented approach with DTLS that uses only lightweight symmetric key operations during the handshake (pre-shared key cipher suite). Note that the use of cipher suites employing public-key cryptography during the DTLS handshake significantly increases the computation overhead. As a result, presented DTLS results correspond to its least expensive case.

We study scalability as a function of the ratio between the total number of DTLS clients (in case of OSCAR, Consumers) and the maximum number of open sessions at a DTLS server. Due to memory limitations, constrained CoAP nodes may have a limited number of DTLS session "slots". We have followed the guidelines on DTLS practical issues (cf. Section 2.1 in the guidelines [54]) and extended the TinyDTLS implementation [111] with the Least Recently Used (LRU) session closure algorithm. The server immediately releases memory and sends a closing alert to the LRU session as soon as a new client has demonstrated good intentions by retransmitting the stateless cookie in the ClientHello message (recall the DTLS handshake). Therefore, the handshake with the new client proceeds immediately. Clients keep their sessions open as long as possible, *i.e.* until they receive the closing alert from the server. In the case of OSCAR, the concept of session slots

does not apply, and the horizontal axis simply denotes the number of Consumers. For example, client/session slots ratio of 16/3 for OSCAR signifies 16 Consumers in the network.

The maximum number of DTLS session slots depends on the platform memory capabilities and actual application memory requirements. With the full IPv6 networking stack of Contiki and a simple application for evaluation purposes, we could have a maximum of 3 session slots on WiSMote (TinyDTLS implementation). However, we should not generalize this number as it depends on the implementation specifics of an application and the operating system. We have used the same number of slots on the GreenNet platform as well to obtain comparable results.

Note that we use two different RDC protocols for the two platforms to test the performance of OSCAR and DTLS running on top of asynchronous (X-MAC / WiSMote) and synchronous (beacon-enabled IEEE 802.15.4/GreenNet) RDC protocols. We set the Beacon Interval of beacon-enabled IEEE 802.15.4 to 122.88 ms to have comparable delays with X-MAC (default channel check interval of 125ms). We summarize the experiment setup in Table 8.1. Simulations in Cooja assumed a star network topology with one Producer (DTLS server) being at the center of the network (it is a radio neighbor for each Consumer and the preferred parent in the RPL DODAG). Note that due to the specifics of beacon-enabled 802.15.4, one node in the network has a mere role of being the PAN coordinator and transmitting periodic beacons. Other nodes in the network associate with it (L2 operation), which effectively introduces an extra hop between Consumers and the Producer, in respect to the network evaluated in Cooja.

Table 8.1: Experiment setup.

(a) GreenNet

Settings	Value
Radio Duty-Cycling	beacon-enabled IEEE 802.15.4
Beacon Interval (ms)	122.88
Superframe Duration (ms)	15.36

(b) WiSMote

Settings	Value
Radio Duty-Cycling	X-MAC
Channel Check Interval (ms)	125
Channel Model	Unit Disk Graph

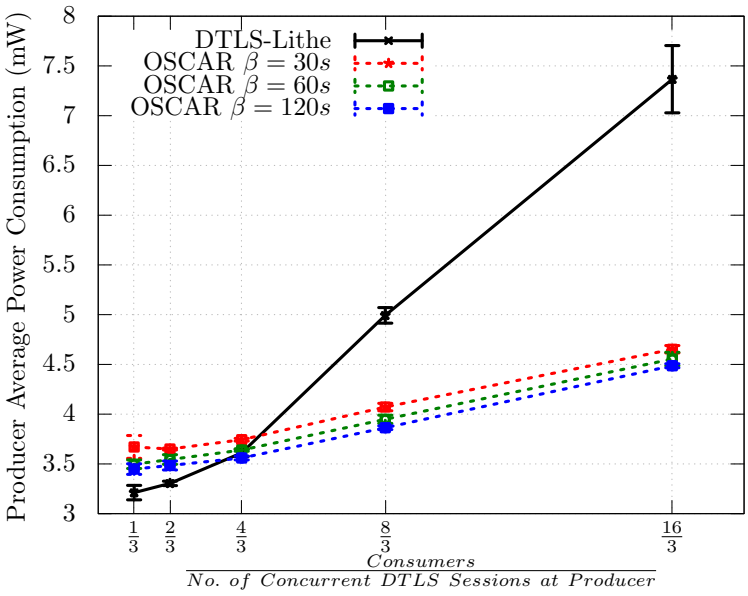
We use a recent 6LoWPAN compression scheme of DTLS named Lithe [127] to maximize its performance. OSCAR Consumers and DTLS clients send a single GET request for a resource on the Producer node (DTLS server) according to the exponential distribution with a mean of 0.5 requests per minute. If the DTLS

session is found open, the request is sent directly without waiting for the handshake to complete. If not, the client first performs a DTLS handshake with the server. Responses contain a resource representation with 25 byte length. In case of OSCAR, this representation is transferred as the appropriate encrypted and signed object type.

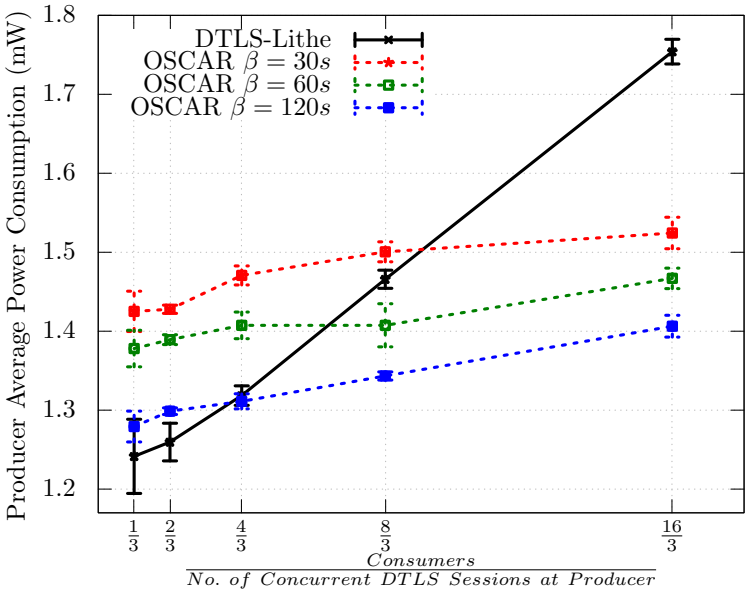
Resource signing load at the Producer is an important aspect of performance evaluation. We define parameter β as the mean re-signing interval such that $\beta = t/N$, where N is the total number of secured resources on the Producer, and t is the average resource update time (for instance, updates of temperature, pressure, CO_2, etc.). We evaluate OSCAR for β values of 30, 60, and 120 seconds, to account for use cases in which high, medium, or low signing load is needed. In the case of OSCAR, we use pre-shared access secrets and certificates to decrypt and verify encrypted and signed objects. Similarly to the work of Hummen *et al.* [61], we use the secp160r1 elliptic curve. Objects are encrypted using the CCM* mode of AES. Similar assumptions apply to DTLS as well: It uses the TLS_PSK_WITH_AES_128_CCM_8 pre-shared key based cipher suite. As a consequence, DTLS only uses symmetric key operations during the handshake. We have run experiments/emulations over 3 hours and plotted 5 run averages with 95% confidence intervals.

Figs. 8.7(a) and 8.7(b) show the impact of the traffic generated by OSCAR on energy consumption. For medium intensity signing load ($\beta = 60s$), in case of WiSMote, OSCAR crosses the energy performance of compressed DTLS when the client/session slot ratio is approximately 1.3. In case of the GreenNet platform, the crossing point increases to approximately 2.15. The crossing points of DTLS and OSCAR curves are mainly influenced by the computation/transmission consumption ratio specific for the MCU/radio transceiver pair. For instance, in the case of WiSMote platform and DTLS with 16 clients in the network, 13.4% of total consumed energy is spent on MCU computations, the rest accounting for radio communication. In case of the GreenNet platform, due to the different MCU/radio transceiver consumption ratio, the percentage accounting for MCU computations increases to 21.2%. For OSCAR and the medium intensity signing load, this percentage increases to 27.9%, due to the heavy utilization of public-key operations and less radio overhead. The results on crossing points are therefore particular for the two evaluated platforms. Nevertheless, the MCU/radio transceiver combinations on the evaluated platforms are very representative – 16-bit Central Processing Unit (CPU) and an old generation radio (WiSMote) and a powerful 32-bit CPU with a prototype low consumption radio transceiver (GreenNet), allowing to estimate the crossing points for a wide range of platforms, between the two demonstrated in our results.

Although our initial design goal was to relieve constrained Producers from traffic and to place burden on Consumers, we can notice in Fig. 8.8(a) that even for client/session slot ratio of 3.7 and 4.17, for WiSMote and GreenNet, respectively, the ECDSA verification results in better performance than using the compressed DTLS approach. Note that in our evaluations, we use constrained Consumers as

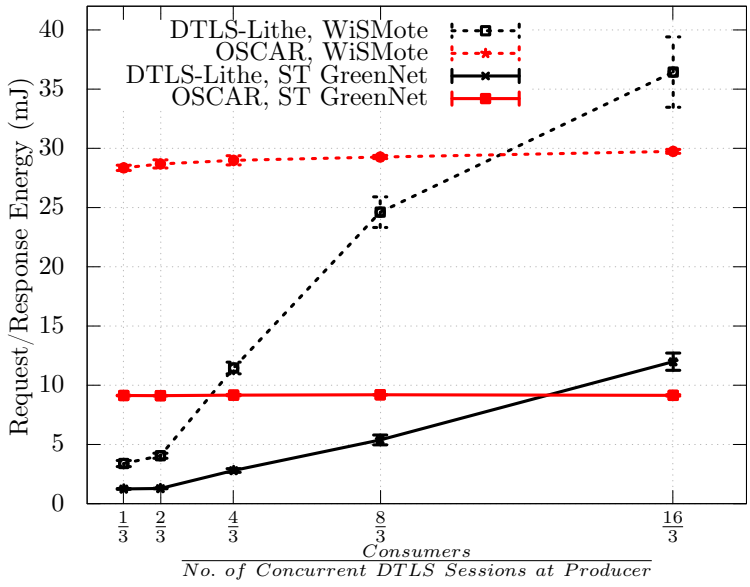


(a) WiSMote

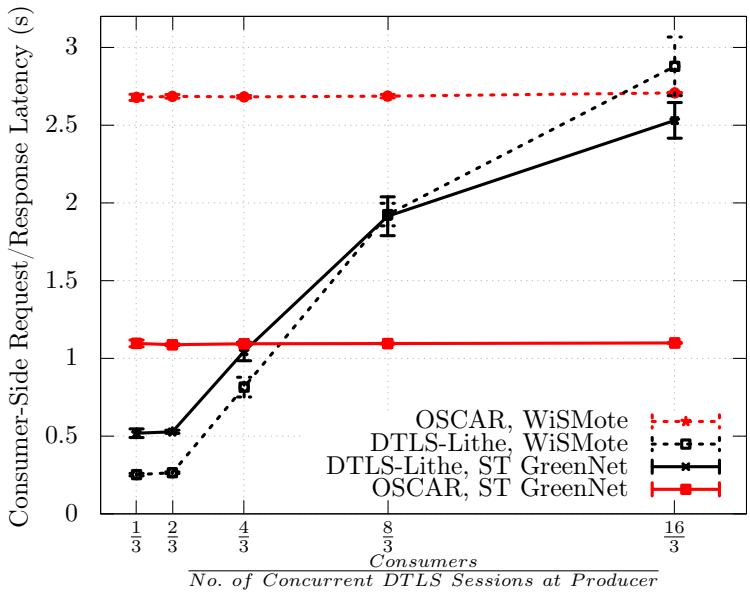


(b) GreenNet

Figure 8.7: Power consumption of a Producer averaged over the experiment time.



(a) Energy consumption



(b) Latency

Figure 8.8: Consumer results per CoAP request-response. They include a possible DTLS handshake.

well thus accounting for the worst case. In IoT use cases, we expect that a significant part of Consumers will include more powerful devices such as smartphones, tablets, laptops, or powerful Cloud servers. From the experimental data, we also obtained an interesting indicator on how OSCAR trades off radio communication at the cost of higher computation load. For example, in case 16 clients are present in the network, a node running DTLS client on WiSMote spends 24.9% of energy during a request-response exchange (possibly including the DTLS handshake) on MCU computations, the rest accounting for radio communication. In the same scenario, OSCAR Consumer spends 83.2% of the energy on MCU computations. With GreenNet platform, the distribution is even more appealing – 20.3% of energy spent on MCU computations with DTLS, and 90.1% with OSCAR.

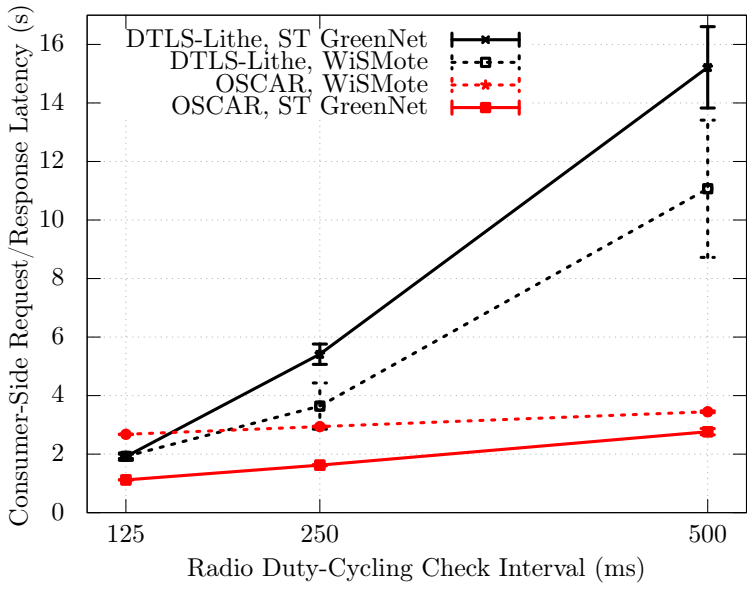
Finally, we evaluate the request-response latency in Fig. 8.8(b). As we can see, MCU computation capabilities greatly affect the result of OSCAR as most of the latency comes from the ECDSA verification. On the GreenNet platform, we have observed an increased number of failed DTLS handshakes for the largest evaluated network with 16 clients due to the stochastic nature of radio links. Note that DTLS curves exponentially increase with the number of clients, but are expected to saturate for denser networks. The exact saturation point depends on the configuration of the DTLS retransmission mechanism (we have used the default retransmission timeout of 2 seconds).

The eventual usage of larger ECC curves will affect the performance results of OSCAR in two main aspects: 1) Increased computation overhead for signing and verification; 2) Radio communication due to the increased length of ECDSA signatures. We believe that aspect 1) will be outweighed by the technological advance in computation capabilities of low-power micro controllers, coupled with increasingly popular adoption of ECC-based hardware accelerators. On the other hand, the usage of larger ECC curves will render larger ECDSA signatures and so the increased per-packet overhead and radio communication. This may be a concern for networks relying on link-layer technologies with very limited frame sizes, such as 802.15.4, as fragmentation threshold may be reached, and thus a mechanism for signature amortization over multiple packets would benefit OSCAR.

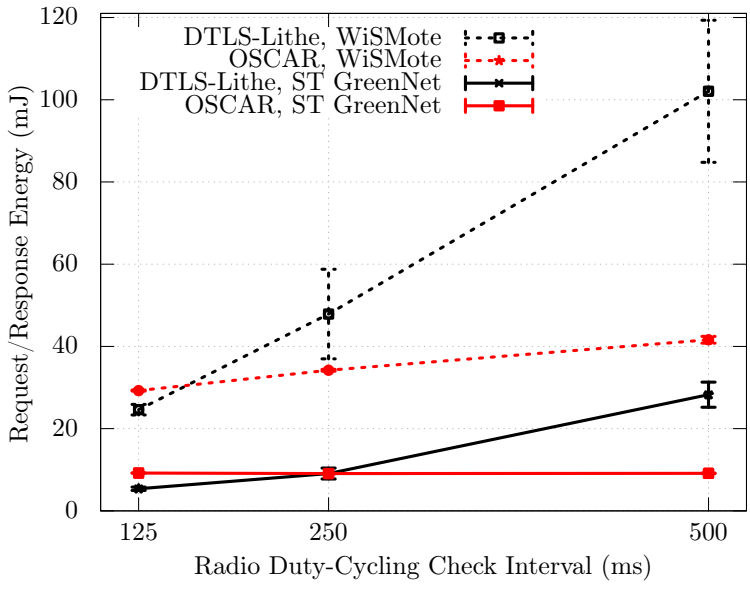
8.6.3 Impact of Radio Duty Cycling

We further demonstrate that the results of OSCAR are agnostic of the duty cycle in the network by evaluating the performance over variable sleep schedules. We study the impact of the main RDC parameters affecting the sleep schedule for the two platforms: Channel Check Rate in the case of X-MAC on the WiSMote platform and the Beacon Interval in the case of synchronous beacon-enabled 802.15.4 on GreenNet nodes. In both cases, the parameters define the periodic wakeup interval of nodes in the network. We have obtained the results in this section for the same traffic pattern as in Section 8.6.2 and the client to session slot ratio of 8/3.

Fig. 8.9(a) presents the total Consumer latency. We can see that the impact



(a) Latency.



(b) Energy consumption

Figure 8.9: Consumer results per CoAP request-response as a function of channel check rate for WiSMote / X-MAC and Beacon Interval for GreenNet / beacon-enabled 802.15.4.

of increasing the wakeup interval on OSCAR is minimal, as no security handshake needs to be performed before the request is sent. Similarly to the result in Fig. 8.8(b), the major part of the latency comes from the processing overhead of signature verification. On the other hand, due to the numerous communications during the handshake, in the case of a lower duty cycle (larger wakeup interval), the DTLS approach results in intolerably high latency. From Fig. 8.9(a), we can conclude that networks with low duty cycles largely benefit from using OSCAR in terms of latency.

The increased latency directly affects the energy consumption as nodes spend more energy on idle listening. As we measure the energy consumption over a request-response exchange, the amount of data that needs to be transferred over the network is independent of the wakeup interval. Therefore, it is not surprising that in Fig. 8.9(b), we observe an increasing energy consumption with the wakeup interval. OSCAR demonstrates a desirable property for constrained networks as the slope of the energy consumption curve is much smaller than that of DTLS. Thus, the results in Figs. 8.9(a) and 8.9(b) show a clear performance advantage of using OSCAR over DTLS even from the point of view of a (constrained) Consumer.

8.7 Impact on Standardization Bodies

Concepts of OSCAR have been widely discussed within IETF, in the context of Authentication and Authorization for Constrained Environments (ACE), DTLS In Constrained Environments (DICE) and IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) working groups.

End-to-end security. In the ACE working group, Selander *et al.* [141] proposed Object Security for CoAP (OSCOAP), a specification for CoAP binding with generic object security format that addresses replay by using sequence numbers within the secured objects. OSCOAP defines specific CoAP fields that can be integrity protected or encrypted end-to-end while traversing proxies. OSCOAP provides two security modes that a user can leverage for different use cases: 1) protection of CoAP header and payload; 2) protection of payload. The main goal of OSCOAP is to provide end-to-end security in the presence of intermediaries (proxies, application-level gateways) using already established keying material. Signaling is done through a newly defined CoAP option. This draft serves as one of the main inputs to the CBOR Object Signing and Encryption (COSE) working group that works on an object security format optimized for constrained devices[1].

Authorization. OSCAR offloads constrained devices from enforcing the authorization decisions by using Access Secrets - group keys that protect confidentiality during transit and allow access to the protected resources. This is useful both when the constrained node is behind a proxy and does not communicate with clients

[1] In the evaluations of OSCAR we used a custom, binary object security format as COSE working group was not formed until April 2015 and the other available options were too heavyweight for constrained devices and IEEE 802.15.4 radios.

directly or when it communicates with a publish-subscribe broker [81]. Publish-subscribe broker receives updates of sensor readings from constrained devices that may sleep extensive periods of time and forwards them asynchronously to subscribed clients. In this context we proposed to ACE a novel authorization flow, called "Client-Pull" [c_9] to handle such scenarios. "Client-Pull" directly spans from OSCAR and the flow presented in Fig. 8.2. We discuss advantages and drawbacks of "Client-Pull" and other authorization flows in the next section.

Object security for network join. In 6TiSCH working group, we discuss the use of object security for the network join protocol. A new node that is not yet a part of the network, needs to reach the gateway that may be multiple hops away and authenticate itself through some shared cryptographic material (a symmetric key or certificate). We consider the encapsulation of the management information, necessary for 6TiSCH join together with replay protection counters, in a secured object. The authentication protocol can then be executed between the gateway and the new node using these secured objects. Similarly to the centralized management of Access Secrets with OSCAR, we are considering to leverage the gateway – Join Coordination Entitiy (JCE) in 6TiSCH terminology – to distribute link-layer keys and offload 6TiSCH nodes of potentially expensive key derivation exchanges.

8.8 OSCAR and Authorization in Constrained Environments

ACE working group in IETF is expected to fill the gaps in compatibility between DTLS and different CoAP features, as discussed in Section 8.1. Ongoing discussions and the ACE charter stating that *"the group is scoped to work only on the web protocols and data carried within them"* suggest that object security may represent an important piece of the final solution(s). In this section, we overview some of the discussed authorization flows, and brief on their advantages and drawbacks for use in constrained environments. The following flows assume a three party protocol with unconstrained Authorization Server (AS), constrained Resource Server (RS) that corresponds to an OSCAR producer, and potentially constrained Client (C) that corresponds to an OSCAR consumer. Client is interested in obtaining access to a resource hosted at the Resource Server, and both Client and Resource Server rely on the Authorization Server that processes access control policies and reaches authorization decisions. We consider out-of-scope how those decisions are reached as they are mainly dependent on bilateral agreements among different service providers [170]. In the following, we focus on abstract communication exchanges that lead to the enforcement of authorization decisions and discuss their advantages and drawbacks for constrained environments.

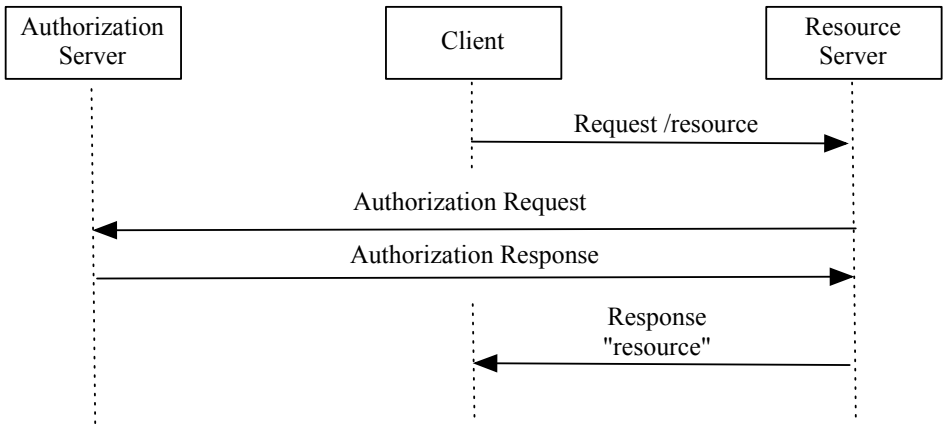


Figure 8.10: Pull scheme.

8.8.1 Pull Scheme

In the Pull scheme (see Fig. 8.10), RS handles the authorization-related exchanges on behalf of the Client. RS contacts AS once it receives a request from the Client and depending on the AS's response, it grants or denies the request.

For RS to handle such a task, it first needs to authenticate C, which in our case corresponds to establishing a DTLS channel. Then, over this secured channel, C sends the necessary authorization information to RS. RS needs to establish another secure connection with AS over which it forwards C's request. Establishing two DTLS channels simultaneously can be demanding for a constrained RS, but a good point is that the channel with AS can be long-lived. Main disadvantage of this scheme is that all the communication burden is put on RS which may lead to easy Denial of Service (DoS) attacks and it requires constant connectivity of RS with AS, which may be outside of the local network.

8.8.2 Push Scheme

Push scheme (see Fig. 8.11) removes the exchanges between AS and RS for each request. When C first contacts RS, it is notified of the corresponding AS that controls RS. Consequently, C can establish a secure, authenticated channel with AS and request authorization for a given resource. AS grants this request by handling to C a *ticket*, that C can present in the next request to RS. Once the ticket is verified by RS, it can respond to the request.

This requires a secure communication channel between RS and C, as C needs to be properly authenticated. A special type of ticket, called *Bearer Token* allows anyone in its possession to access a resource [67], but requires to be exchanged over a secure channel. The main concern with this scheme is that RS needs to perform an expensive DTLS handshake with each client.

OSCAR with PUT/POST/DELETE requests corresponds to the Push scheme. Instead of handling a special ticket or token, OSCAR AS hands to C a cryptographic

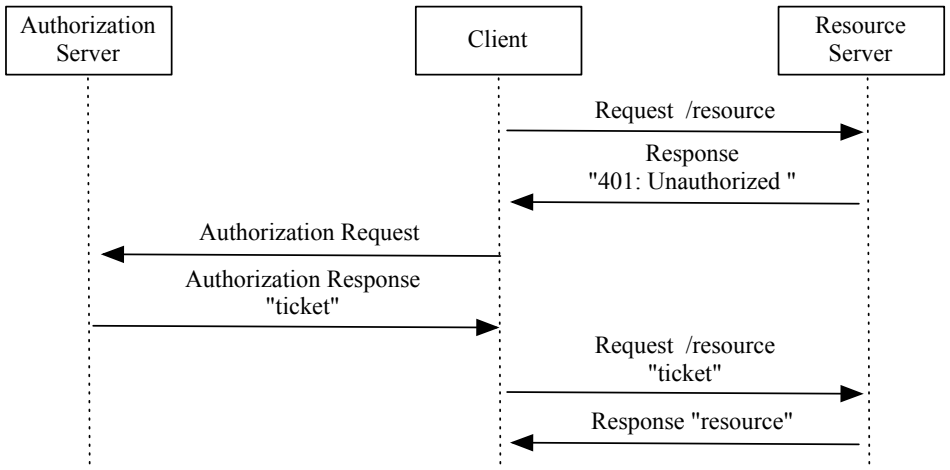


Figure 8.11: Push scheme.

key, Access Secret, that C can use to both replay protect and prove its access rights. In the web terminology, the OSCAR approach corresponds to Proof of Possession authorization tokens. In this case, communication between C and RS does not need to be performed over a secure channel, which significantly offloads RS. This is also convenient for use cases where AS can be offline, such that C cannot contact AS for each request. The downside of our approach is that each Client possesses the Access Secret. Once the access rights of a given client are revoked, AS needs to inform RS of the new Access Secret. Correspondingly, the remaining authorized Clients will need to contact AS for the new Access Secret in order to perform new requests.

8.8.3 Client-Pull Scheme

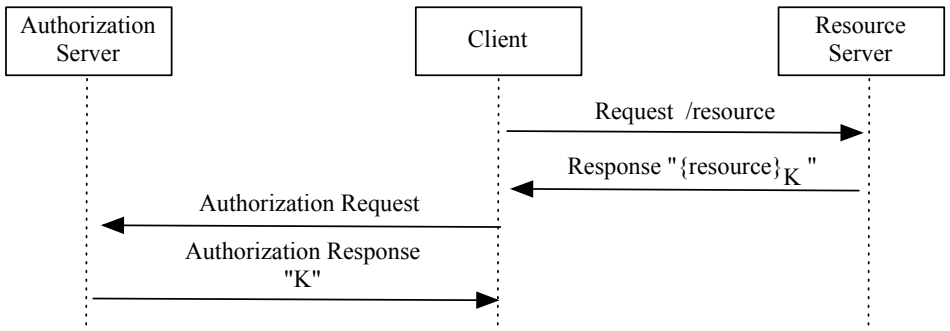


Figure 8.12: Client-Pull scheme. $\{X\}_K$ denotes encryption of X with key K.

Finally, from OSCAR we derived a Client-Pull scheme (see Fig. 8.12) for GET requests[2] [c_9]. Client-Pull completely offloads RS from enforcing authorization decisions by giving unconditional access to *protected* resource representations. These

[2]This includes asynchronous notifications, i.e. convergecast traffic.

protected resource representations contain enough information for C to contact AS. AS enforces authorization decisions by sharing with authenticated Clients cryptographic keys, i.e. Access Secrets, that can be used to access the resources.

Our constrained RS is exposed to every C. This is necessary to handle the use cases when AS is offline. Communication between RS and C, however, does not require a secure channel and expensive DTLS handshake. We saw in Chapter 4 that symmetric encryption is negligibly cheap and much preferable to radio exchanges. Note that sending a response "401: Unauthorized" involves a radio transmission, as expensive as sending the protected resource due to the small size of sensor readings.

The obvious concern is DoS, which can be handled by throttling the response rate at RS. Note that such DoS attack can also be performed using DTLS by initiating fake handshakes that never complete. A second security concern is that each response triggers a replay counter increment and an attacker may be able to force RS to reuse the key[3], which would break confidentiality. Our analysis in Section 8.4 considers this case and gives an estimate how soon such an attack can cause damage, given λ, the request rate. Additionally, RS and AS can agree on a protocol that will allow RS to request a new Access Secret, once it detects that the counter has wrapped.

This scheme is also applicable to any use case where RS communicates with different Clients through intermediary. One example of such intermediary may be a CoAP reverse proxy [144]. Another example is a publish-subscribe broker [81]. Last but not the least is a generic application-level gateway present in GreenNet architecture and also extremely common in existing Wireless Sensor Network (WSN) deployments [42]. In those cases, RS communicates with a single node, which additionally minimizes the discussed security concerns.

8.9 Conclusion

Our work explores a novel approach to the problem of end-to-end security in IoT. It is based on the concept of object security that relates security with the application payload.

In the proposed OSCAR architecture, we move expensive radio communications from constrained CoAP nodes to more powerful servers. We introduce Authorization Servers that store the certificates and provide Access Secrets to Consumers to enable them to request resources from Producers.

The scheme separates confidentiality and authenticity trust domains. Confidentiality is used as a means to provide capability-based access control and a protection against eavesdropping during the communication. The scheme allows source authentication and the trust in the content generated by Producers. In turn, this property enables local databases and caches to use the secured content. Moreover, leveraging the access right confidentiality domain and the concept of object security,

[3] In a different setting from Eq. 8.1, this would equivalent to reusing the Initialization Vector (IV).

our proposal intrinsically supports multicast. We take off the burden of a security handshake with every client from constrained nodes. Instead, we rely on secure communication channels with Authorization Servers that are in charge of resource access right key management. Cryptographic burden shifts to Consumers that need to perform signature verifications for the content they are interested in.

We have demonstrated the feasibility of the proposed architecture by evaluating its performance in several scenarios even for the highly constrained case of M2M communications on two hardware platforms. The results show that OSCAR outperforms a security scheme based on DTLS when the number of nodes increases. OSCAR also results in low energy consumption and latency.

Part III

Conclusions

Chapter 9

Lessons Learned and Future Directions

Technologies that enable the Internet of Things are part of our reality for years. First standards were published as of 2003, and the best example of technological readiness is the level of attention security is getting today in the standardization world. Products are shipping and few pieces are missing to enable the ubiquitous connectivity and the most disruptive innovation since the emergence of the Internet. Those pieces are interoperability and energy-efficiency. But tackling one without the other is the path to either unsatisfactory products and frustrated users or to the world of proprietary solutions we witnessed during the last decade.

9.1 Summary of Results

Our research explored the intersection of academic, industrial and standardization spheres because we believe that advancing state-of-the-art for the public benefit necessitates the convergence of all three. We apply the existing standards to constrained devices of the Internet of Things and draw conclusions on their applicability, performance and potential gaps.

In that context, we start from the constrained hardware and implement and evaluate the fundamental cryptographic primitives. We observe that hardware-accelerated cryptography is a must for Internet of Things devices, as it leads to reductions in execution time, as much as two orders of magnitude. We study in details the origins of the computational overhead and conclude that there are many benefits to implementing full block-cipher Mode of Operation in hardware, due to the high I/O access latencies, that account for almost half of the total overhead, even in case of System on Chip architectures. With these results in mind, we contribute an Application Programming Interface that reduces the development time and maximizes the performance, leveraging the available hardware, together with implementations for three Internet of Things devices.

We pass from a single device to the wireless network by studying the cost of communication security that protects the local area network from radio-range attackers. Overhead of the cryptographic primitives is only one of the factors that influences the overall performance in the networking context. To understand the energy – security tradeoffs, we practically evaluate the effect of link-layer security features on the performance of Wireless Sensors Networks and revisit the contradictory conclusions found in the literature. We discuss that many real world, security

agnostic factors affect the energy consumption of a device, and show that this leads to exaggerated conclusions on the energetic cost of security. We show that for practical applications and implementations, link-layer security features introduce a negligible degradation on the order of a couple of percent, that is often acceptable even for the most energy stringent systems, such as those based on energy harvesting. Similar conclusions hold for Time-Slotted Channel Hopping systems where timings are critical – hardware acceleration of cryptography is simply a necessity, and the performance of hardware acceleration blocks largely affects the minimum achievable latencies and throughput in the network.

Because link-layer security puts trust on each node on the communication path that consists of multiple, potentially compromised devices, we protect the information flows by end-to-end security mechanisms. Furthermore, as we move away from the local network, our IP-enabled constrained devices may be faced with an adversary that may be located anywhere in the Internet. We therefore consider Datagram Transport Layer Security (DTLS) protocol, the IETF standard for end-to-end security in the Internet of Things and contribute to the debate in both the standardization and research communities on the applicability of DTLS to constrained environments. The main concerns are the communication overhead and latency of the DTLS handshake, and the memory footprint of a DTLS implementation. We provide a thorough performance evaluation of DTLS in different duty-cycled networks through real-world experimentation, emulation and analysis. Our results demonstrate surprisingly poor performance of DTLS in networks where energy efficiency is paramount. Because a DTLS client and a server exchange more than 10 signaling packets, the DTLS handshake takes between a handful of seconds and several tens of seconds, with similar results for different duty cycling protocols. Moreover, because of their limited memory, typical constrained nodes can only maintain several simultaneous DTLS sessions, which highlights the need for using DTLS parsimoniously.

Apart from its performance issues, DTLS was designed for point-to-point communication dominant in the traditional Internet. The novel Constrained Application Protocol (CoAP) was tailored for constrained devices by taking into account requirements such as asynchronous application traffic, group communication and absolute need for caching. These requirements do not disappear when security is considered in the picture. The security architecture based on DTLS is however, not able to keep up and advanced features of CoAP simply become futile when used in conjunction with DTLS. We therefore propose an architecture that leverages the security concepts both from content-centric and traditional connection-oriented approaches. We rely on secure channels established by means of DTLS for key exchange, but we get rid of the notion of "state" among communicating entities by leveraging the concept of object security. We provide a mechanism to protect from replay attacks by coupling the capability-based access control with network communication and CoAP header. OSCAR, our object-based security architecture, intrinsically supports caching and multicast. Moreover, it does not affect the radio duty-cycling operation of constrained devices. We demonstrate the benefits of

OSCAR for Machine to Machine communication, two different hardware platforms and duty-cycle protocols on a real testbed and using an emulator. We show significant energy savings at constrained servers and reasonable delays. Ideas from OSCAR have already found their way towards the Internet standards and are heavily discussed as potential solutions for standardization.

Apart from the challenges of securing IP-based Wireless Sensor Networks, we also tackle two problems related to the construction and maintenance of the network. First, motivated by incompatibilities of two prominent Internet of Things standards, we propose a new scheme that allows coupling *beacon-enabled* IEEE 802.15.4 with RPL, the Internet standard for routing in WSNs. Second, we analytically model the mechanism that controls the emission of network maintenance packets with RPL – the Trickle algorithm. We demonstrate unfairness that may arise in different network topologies, that leads to unbalanced transmission load in the network, when using Trickle. We present these results in the Appendix.

9.2 Evolutions of OSCAR and Future Perspectives

We first worked on OSCAR in the first quarter of 2013 when CoAP specification [144] was still a draft. As such, our design made an (optimistic) assumption that it was possible to re-define CoAP in order to meet different security requirements. The best example is `MessageID` field that is used for duplicate detection of CoAP messages that we decided to use as a proof-of-concept replay protection mechanism. This was not necessarily a good choice as CoAP never intended this field to be used for security purposes and implementations based upon it would therefore not be secure. Latest efforts [141] simply shift the replay protection within protected objects by including sequence numbers as one of the fields. However, we intended to demonstrate an important concept that still holds whether replay protection is based on a CoAP field or any specific mechanism within the protected objects – *separation of communication security from data security*. We achieved this by protecting with outermost encrypted objects communication-related fields, which leaves the inner object(s) agnostic of network communication and facilitates off-line usage.

We discussed the necessity that the outermost encrypted object should protect CoAP header from altering, in order to prevent semantic attacks on CoAP applications. In group communication cases or when one Access Secret is shared on multiple nodes, OSCAR on its own cannot protect against an insider attack. On one hand, OSCAR provides means to integrate crypto mechanisms that are resilient against such an attacker [153] with the network architecture in a clean way. On the other hand, if malicious nodes can be detected by other means, one may simply update the Access Secret and exclude the malicious node from the group using Access Secret (key) management protocol [20, 168]. OSCAR is in that sense a placeholder for such protocols/schemes and facilitates their adoption by defining an architecture that accounts for multiple traffic patterns, specific for, but not only applicable to

Internet of Things.

Another interesting research direction we want to pursue is the integration of multicast authentication protocols [118, 21] in order to avoid the usage of expensive, digital signatures while keeping the separation of confidentiality and authenticity trust domains.

In summary, a fully-deployable solution based on OSCAR is composed of three main components:

1. Binding of object security with CoAP for end-to-end secure communication between a Producer and Consumer(s).

2. Protocol that runs among Producers, Consumers and Authorization Server(s) to bind resources with different Access Secrets and enforce authorization decisions by sharing those Access Secrets with appropriate Consumers.

3. Access Secret management protocol.

Points 1 and 2 are, at the time of the writing, being actively tackled in IETF [141] [c_9], and represent one of the standard candidates within the ACE working group. Point 3 does not affect interoperability and we therefore expect proprietary schemes to be present in different deployments, depending on the application security requirements. Nevertheless, we plan on adapting the different group key management schemes proposed in the literature to the context of OSCAR.

Finally, OSCAR concepts are fully applicable outside of the constrained space of Internet of Things. We believe that object security is the way towards the content-centric security of the Web, and its facility of encapsulation a perfect means for protecting our privacy.

Appendix
Energy-Efficient Construction and Maintenance of the Network

Appendix A

Standards-Based Incompatibilities

In this part, we take a step back from Internet of Things (IoT) security challenges and consider the local operation of an energy-constrained network. In that context, we tackled two problems that stood on the way of seamless integration of IEEE 802.15.4-based Wireless Sensor Networks (WSNs), such as GreenNet, with the Internet standards and so the Internet infrastructure. To be applicable to a wide range of settings, standards often need to be sufficiently generic but this comes at a performance price that may not always be acceptable. One example is the interaction of the Internet standard for routing in WSNs with *beacon-enabled* mode of IEEE 802.15.4, used on GreenNet nodes. Energy-constraints of GreenNet are so severe that every exchanged packet in the network counts and only the most necessary information need to be exchanged. Incompatibilities between abstraction layers often lead to such unnecessary exchanges. For this reason, the GreenNet team designed a proprietary routing protocol [85] that would specifically fit the needs of GreenNet nodes.

Motivated by the underlying issues of such incompatibilities, we have explored the path of standard-compliant optimizations. The two contributions presented in this part are directly or indirectly motivated by IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) – the Internet Engineering Task Force (IETF) standard for routing that would allow WSN to be only a small cluster in the much greater Internet routing infrastructure. First, in Appendix B we explore the problematics of running RPL on top of *beacon-enabled* IEEE 802.15.4, to provide a means for GreenNet nodes to run the Internet standard. Second, in Appendix C we study the crucial component of RPL for controlling the number of exchanged packets in the network – the Trickle algorithm. As Trickle is a widely used algorithm, we do so in a generic manner that is applicable to any of its applications, not just the RPL.

In the remaining of this chapter, we present the necessary background for this appendix. Section A.1 summarizes the functioning of RPL and is mainly an adaptation of the corresponding section in the author's Master's thesis. We overview the functioning of the Trickle algorithm, in Section A.2 as its functioning is necessary for comprehension of both Appendix B and C.

A.1 Routing Protocol for Low-Power and Lossy Networks

IETF formed a Routing Over Low-Power and Lossy Networks (ROLL) working group in 2008 that conducted an analysis of routing requirements for typical IoT applications, such as home, building and industrial automation, and urban networks including smart grids [171]. The question posed was whether any of the existing routing protocols satisfy the IoT requirements. The conclusion of the study [92] was that a new protocol is required and that it should not make any assumptions on the underlying link layer, due to the wide variety of IoT link-layer technologies: IEEE 802.15.4, low-power IEEE 802.11, Power Line Communication (PLC) using IEEE 802.15.4 such as IEEE P1901.2.

The outcome of ROLL standardization is IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL). RPL was designed with a proactive approach in mind – the routes are found and maintained without any considerations on the ongoing traffic in the network. A Directed Acyclic Graph (DAG) is formed over a mesh network by specifying how link costs and node attributes need to be combined to compute paths costs.

In essence, RPL is a distance vector protocol that specifies how to construct a Destination Oriented Directed Acyclic Graph (DODAG) with a defined objective function and a set of metrics and constraints. An example of a DAG and an appropriate DODAG is shown in Figure 1.1. Several DODAG instances may be used for the same mesh network which allows for traffic differentiation in classes. For instance, high priority traffic could use the minimal-delay path through the network while low priority traffic could avoid battery-powered nodes and be routed along a path consisted of mains-powered devices.

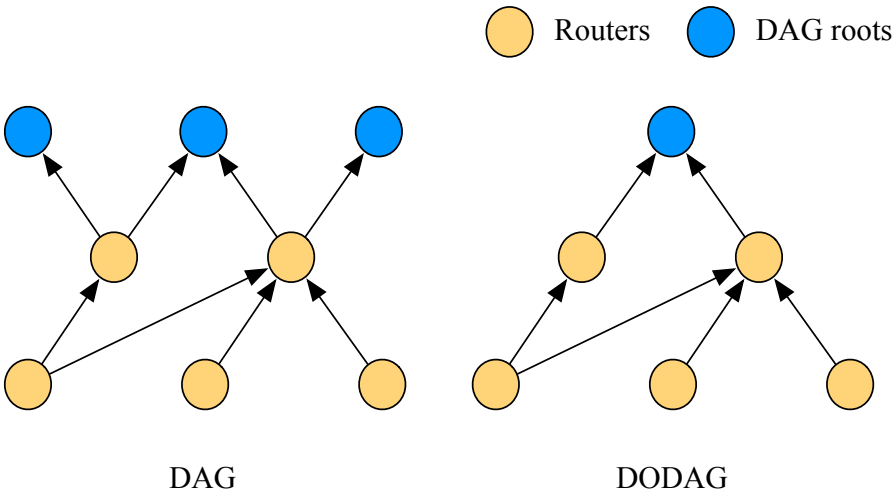


Figure 1.1: An example of a DAG terminated at sink nodes and a possible DODAG.

In order to define and maintain a topology, RPL uses four main identifiers:

- `RPLInstanceID`, identifier of a specific RPL instance within the network. Each instance can serve different constraints and performance criteria.
- `DODAGID`, specifying one DODAG within a `RPLInstance`.
- `DODAGVersionNumber`, identifier used within the network in order to monitor changes in the topology. It is incremented each time DODAG is rebuilt.
- `Rank`, identifier of a position of a node in respect to a DODAG root. The rank must monotonically increase as the DODAG is followed towards the leafs. The exact calculation depends on the objective function used.

The RPL routing protocol specifies a set of new Internet Control Message Protocol version 6 (ICMPv6) control messages to exchange information related to a DODAG:

- **DODAG Information Object (DIO)** defines and maintains upward routes to the root, i.e. the DODAG.
- **DODAG Information Solicitation (DIS)** message is used by a node in order to pro-actively solicit DODAG related information from neighboring nodes. They are typically transmitted when a node first joins a network.
- **DODAG Destination Advertisement Object (DAO)** messages are used to advertise prefix reachability towards the leaf nodes of the DODAG. They enable traffic to flow also in downward direction – from the root towards the leaves.

The root[1] starts the DODAG building process by transmitting a DIO message. Neighboring nodes will process DIO messages potentially from multiple nodes and make a decision on joining the DODAG based on the objective function and/or local policy. A node has a route towards the root as soon as it joins the graph. The node computes its `Rank` in respect to the root and starts advertising DIO messages to the neighbors, with updated information. As the process converges, each node in the network will have received one or more DIO messages and will have a preferred parent selected. The preferred parent is therefore used as the next hop on the route towards the root. The RPL protocol optimizes the upward routes for convergecast traffic, as it is the dominant traffic pattern in WSNs.

In order to support downward routes, RPL uses DAO control messages that are addressed as unicast packets and sent upwards. DAO messages describe prefix information, route lifetime and other information about the distance of the prefix. RPL defines two modes in which a network can operate in respect to the management of downward routes:

[1] Although the root is most commonly the Personal Area Network (PAN) coordinator (WSN sink), it is possible to have it outside of the local network, such that the WSN is merely a subnetwork of an infrastructure with various link-layer technologies.

- **Storing mode** – where each node keeps track of all downlink prefixes that are accessible through it. Node learns the accessible downward prefixes after processing DAO messages from its children and advertizes them upwards through its parents, towards the root. This requires all the routers to maintain a routing table with downlink entries which may be challenging due to the memory limitations of constrained devices.

- **Non-storing mode** – Each node unicasts a DAO message containing one or more of its parents towards the root of the DODAG. Intermediate routers do not keep any state, just forward the packets. The root will eventually receive a DAO from each node in the network and will therefore be able to construct the downward routes. The root can then transmit a message towards a given node by using source routing, where each intermediate hop towards a specific destination is signaled in the header of a packet.

DIO and DAO control messages are used to enable multipoint-to-point (convergecast) and point-to-multipoint (root towards sensors) communication, respectively. Point-to-point communication is enabled as a combination of the two mechanisms. A packet destined towards a certain node in the network, in storing mode, will travel up to the common ancestor in the DODAG, from where it will be forwarded downwards. In the case of non-storing mode, the packet will travel all the way up to the DODAG root which will then forward the packet downward using source routing. This mechanism is far from optimal but RPL assumes that such occurrences are rare.

Emission interval of DIO control messages, and therefore the control overhead of RPL, is regulated by the Trickle algorithm [91]. The idea is to reduce DIO emissions by transmitting less frequently when there is no change in the topology.

A.2 The Trickle Algorithm

The main idea of the Trickle algorithm is on one hand to exponentially reduce the amount of control traffic in the network, while there are no detected inconsistencies. On the other hand, once an inconsistency has been detected it quickly propagates the new information state. Naturally, the ”consistency notion” is defined by the protocol or application actually using Trickle. For instance, in the case of RPL, consistency is checked by comparing the advertised DIO state in the network to the local one. Trickle was originally designed for firmware versioning in WSNs [91]. In this case, consistency is checked by comparing the advertised and local software versions.

Trickle splits time into intervals of variable length where transmissions may occur following Trickle's rules. The three parameters to configure Trickle are: i) I_{min}, the minimum interval size; ii) I_{max}, the maximum interval size expressed as the number of times the minimum interval may double; iii) K, the redundancy constant.

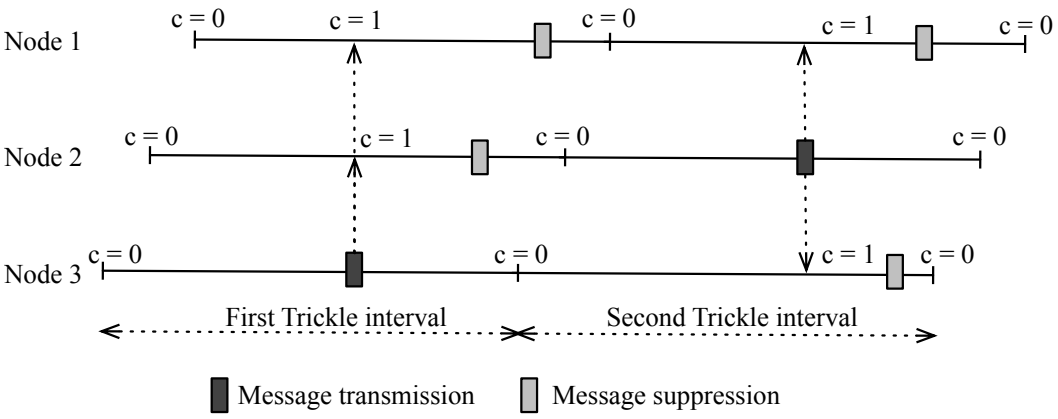


Figure 1.2: Example of Trickle algorithm in steady state with the lack of synchronization among nodes. Redundancy constant $K = 1$, and all nodes are neighbors.

A node following the Trickle algorithm increments a local counter c for each consistent reception. The node transmits at instant t if:

$$c < K, \tag{1.1}$$

that is, if the number of consistent receptions is smaller than the redundancy constant. Counter c is reset to zero at the beginning of each interval. Instant t at which Trickle decides if it is going to transmit is selected randomly from the uniform interval $[\frac{1}{2}I, I)$, where $I \in \{I_{min} \times 2^n \mid n \in \mathbb{N}_0, n \leq I_{max}\}$. Interval I is doubled upon its expiration by incrementing n. When a node detects inconsistency, n becomes 0, which sets interval I to I_{min}. Fig. 1.2 illustrates an example Trickle operation in steady state for $K = 1$. As soon as $c \geq K$, transmissions are suppressed. Note that Trickle intervals among nodes are not necessarily synchronized.

Appendix B

Topology Construction in RPL networks over Beacon-Enabled IEEE 802.15.4

B.1 Introduction

In this chapter, we address the problem of running the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [178], the IETF standard for routing in Wireless Sensor Networks (WSNs), on top of IEEE 802.15.4 *beacon-enabled* nodes [c_8].

The forwarding structure built by RPL is a Destination Oriented Directed Acyclic Graph (DODAG). Each node keeps a list of available parent nodes closer to the DODAG root and selects one of them as the "preferred parent" based on an objective function. When a link to the preferred parent fails, a node switches to another parent in its list. At the link layer, the *beacon-enabled* IEEE 802.15.4 nodes need to construct a cluster-tree anchored at the Personal Area Network (PAN) coordinator (also the sink node) for supporting multi-hop communication. Moreover, a node joining the cluster-tree has to associate with a coordinator (a Layer 2 operation) before it may send any data frame. The choice of the coordinator influences any possible choice of the RPL parent node. In the case of the *beacon-enabled* IEEE 802.15.4 nodes, the problem is how to construct the IEEE 802.15.4 cluster-tree according to the RPL routing information based on a DODAG.

While both *beacon-enabled* IEEE 802.15.4 and RPL have been extensively studied within their abstraction layer, the joint operation is surprisingly still an open problem. The existing work in the literature [116] requires extensive modifications to both standards, which is an unrealistic requirement at the current stage of Internet of Things (IoT) stack development.

We propose a solution to the problem that satisfies the constraint of keeping RPL and IEEE 802.15.4 unchanged. In our approach, RPL constructs its DODAG before the cluster-tree at link layer and we use the RPL routing information (selection of the preferred parent) in the association decision to establish links, i.e., to select the coordinator in the cluster-tree that is the preferred parent in the DODAG.

The proposed solution takes advantage of cross-layer signaling: a node joining the network requests RPL information from neighbor IEEE 802.15.4 coordinators and associates with the right coordinator based on the information in a RPL message. We adapt the operation of the Trickle timer [91] that governs the transmission

of RPL messages to provide the required information to the link layer (the adaptation remains compliant with the RPL specification).

The main contributions of this chapter are the following:

- a new scheme that allows RPL to run over the *beacon-enabled* IEEE 802.15.4 without any modification to the two standards,
- the scheme leading to energy savings both during the topology construction and in the steady-state, due to the use of the Trickle timer,
- a simple probabilistic model of the Trickle timer and an analysis of the delay of the proposed scheme,
- an evaluation of energy savings and the time for topology convergence based on the implementation of the proposed scheme in Contiki.

The remaining of the chapter is organized as follows. We describe the problem of running RPL over *beacon-enabled* IEEE 802.15.4 in Section B.2. We provide a detailed description of the proposed scheme in Section B.3 and evaluate it in Section B.4. Section B.5 summarizes the related work and Section B.6 concludes the chapter.

B.2 Forming the Cluster-Tree in Beacon-Enabled Mode

As we could see in Chapter 3, the operation of nodes in the *beacon-enabled* mode of IEEE 802.15.4 relies on beacons that delimit the start of a *superframe*. Immediately following is the Contention Access Period (CAP) during which nodes transmit pending data frames to their parent (cluster coordinator) using the slotted Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) algorithm (a coordinator node needs to stay active during CAP). Beacon Order (BO) and Superframe Order (SO) are the key parameters to tune the desired level of radio duty cycling in the *beacon-enabled* mode and the relation between them is given in Eqs. 3.1 and 3.2.

The network formed in the *non-beacon* mode may be a mesh in which each node may communicate with its radio-range neighbors, so running RPL in this case does not raise any problems. Nodes in the *beacon-enabled* mode have to form a *cluster-tree*: a node selects one parent node, the cluster coordinator, and synchronizes with its beacons. The node may become a coordinator itself on behalf of other nodes, which enables multi-hop communication from leaf nodes to the root of the cluster-tree.

The PAN coordinator is the root of the tree, the sink of the sensor network. It starts the topology construction by transmitting the first beacon. Other nodes are unassociated and have to switch their radio transceivers on to perform passive scanning, the only mechanism for discovering potential coordinators available in the *beacon-enabled* mode. The reception of a beacon initiates a scan period during

which a node waits for beacons. At the end of this period, a node can initiate the association with the best coordinator with the sequence of `association-request`, `ack`, `data-request`, `association-reply`, `ack` control frames.

Note that most of the energy consumed during the topology construction phase comes from idle listening during the scan period, which is unavoidable for any association strategy that discovers the best available coordinator. The duration of this interval should allow the discovery of all coordinators in the radio range.

Fig. 2.1 illustrates a timeline of the topology construction for an example cluster-tree composed of four nodes. Note that Node 4 may receive beacons from Coordinators 2 and 3, but it selects Node 3 as the best parent.

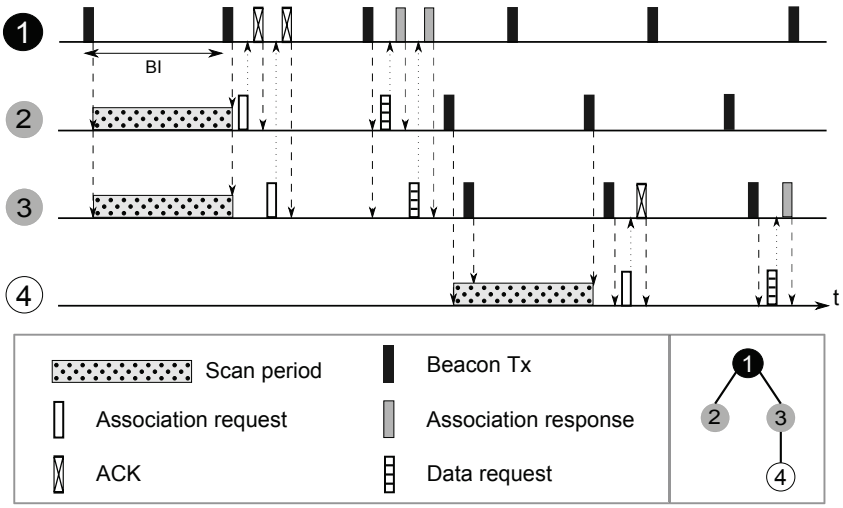


Figure 2.1: Topology construction in an example 802.15.4 cluster-tree.

Incompatibilities with RPL. As we could see in Appendix A, RPL [178] is a Distance Vector protocol that specifies how to construct a DODAG with a defined objective function and a set of metrics and constraints. In case of *beacon-enabled* IEEE 802.15.4 at the link layer, the traditional layer-independent operation would confine the selection of RPL routes to those in the already constructed L2 cluster-tree. Consequently, the overall performance of RPL would be significantly degraded. We exploit the approach of merging two structures: the 802.15.4 cluster-tree and the DODAG of RPL, which allows us to benefit from low overhead, small delays, and near optimal upward routes of RPL [161] while creating the IEEE 802.15.4 cluster-tree required for low duty cycle communications.

B.3 802.15.4 Cluster-Tree Construction Based on RPL DODAG

We propose the selection of the best coordinator in the 802.15.4 cluster-tree based on the preferred parent in the DODAG of RPL. The resulting cluster-tree will

effectively be a subset of the DODAG initialized during the topology construction phase. There are several issues with such an approach:

1. RPL is a network-layer protocol, but no communication among nodes at the network layer may take place before links at the link layer are established (node association with a coordinator).

2. An IEEE 802.15.4 node once associated can only communicate with its cluster coordinator, so after association, a node can only receive DODAG Information Object (DIO) messages from its cluster coordinator.

To address the first issue, we exploit the fact that DIO messages are multicast. As network-layer multicasts translate to link-layer broadcasts, we use beacons to broadcast DIO messages. There is no better broadcast mechanism in multihop *beacon-enabled* networks than the beacons themselves—during the scan period devices wait for beacons. We assume that IEEE 802.15.4 Reduced Function Device (RFD) is configured as RPL leaf node, i.e., it does not send DIO messages. Similarly, Full Function Device (FFD) may become cluster coordinator, i.e., has to be configured as RPL router, which is a realistic assumption as the role of a device mainly depends on its energy source. We assume that a node a priori knows if it is an RFD or an FFD.

We propose the encapsulation of RPL DIO messages in the beacon frame payload following an idea discussed in the team [2]. Link layer adds DIO to the payload of the next scheduled beacon if the resulting frame does not exceed IEEE 802.15.4 Maximum Transmission Unit (MTU) of 127 bytes (cf. Fig. 2.2). In case the DIO message cannot fit into the current beacon, it may be fragmented or delayed for the following one as the beacon payload size varies as a function of downward traffic.

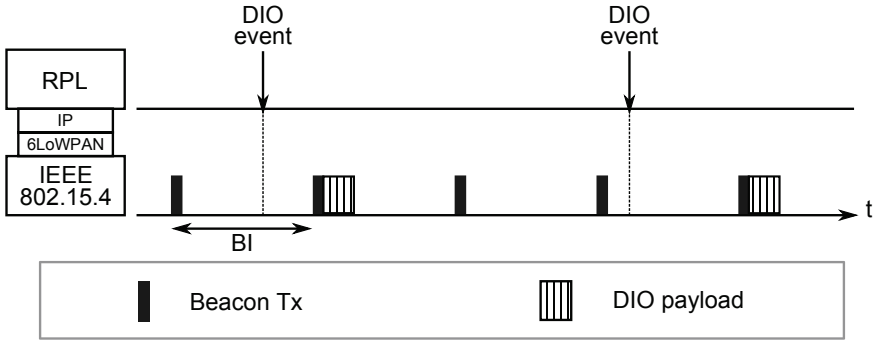


Figure 2.2: Encapsulation of DIO messages in beacon frames.

The exponential increase of the DIO transmission interval governed by Trickle has an important side effect: arriving nodes would potentially wait a long time interval before receiving the first DIO. RPL addresses this issue with DODAG Information Solicitation (DIS) messages that can be broadcast to solicit the transmission of a DIO: upon reception of a DIS, a node resets its Trickle interval I to I_{min} so DIO will be transmitted shortly [178]. However, DIS broadcast is not

enough for synchronous duty cycled networks—neighbor nodes in the radio range may sleep at the instant of the DIS transmission. As explained above, the reception of a beacon delimits the start of the CAP during which the coordinator is active. Thus, CAP is the most suitable period during which an unassociated node may solicit information from nearby coordinators. Note that a node wanting to join the network is awake during the scanning period so it can receive beacons from several neighbor coordinators. Thus, we propose that the node transmits a solicitation message by performing CSMA/CA after the beacon if the following two conditions hold:

- the received beacon is the first one received from a given coordinator,
- the beacon does not contain a DIO in its payload.

The solicitation message could be a RPL DIS message encapsulated in an IEEE 802.15.4 command frame. Note that a node cannot send data frames before association [62]. However, we have chosen to use the IEEE 802.15.4 *beacon-request* command frame without any payload as a solicitation message—it has a small size (8 bytes) so a very short transmission time. Additionally, the RPL specification [178] allows the Trickle reset triggered by external events.

Note that the *beacon-request* command frame is typically used in the *non-beacon* mode to solicit the information about the network. It has no use in the *beacon-enabled* mode as beacons are periodically transmitted. We use its reception at link layer to trigger the reset of the Trickle timer at the RPL layer to spawn a DIO transmission. The goal is to encapsulate the DIO message in the following beacon so that arriving nodes can select the best coordinator. As a node may send several *beacon-request* solicitation frames during the scan period (and CAP of each detected coordinator), the scheme ensures the reset of the Trickle timer for all RPL routers in the range.

A possible drawback of the scheme could be its possible side effect on the duration of the always-on scan period. In fact, with typical parent selection schemes at link layer, each beacon carries a network-specific metric processed by arriving nodes. Then, in case BO is a priori known, the worst-case scan duration is one Beacon Interval (BI). However, a simple algorithm achieves the same duration with our scheme as well—during the scan period of duration BI:

1. for each discovered coordinator, a node stores the expected instant of the next beacon ($current\_time() + BI$),
2. for each discovered coordinator, a node solicits the reset of the Trickle timer as explained above,
3. upon expiration of BI, a node goes to sleep and schedules its wake up at the instants found in (1),
4. a node wakes up and receives the beacon with the DIO payload,

5. upon reception of the DIO payload from the last discovered coordinator, a node consults RPL about the best choice and schedules the next wake up just before the beacon of the selected coordinator; then, the node follows the standard association procedure.

This scheme ensures the discovery of all coordinators in the radio range while allowing a node to start duty cycling after one BI from the boot time (cf. Fig. 2.3). During next BI, node receives DIOs and passes them to RPL. In the worst case, by the end of the second BI, RPL will have the preferred parent selected. The additional *worst-case* delay of one BI is the price to pay during the topology construction for the benefit that comes later-on with the Trickle timer during the network operation. As the node spends most of the second beacon interval sleeping, it consumes energy only for receiving beacons. For n discovered coordinators, the energy will be $E = n \times T \times I_{RX} \times V$, where I_{RX} is the radio current draw in receive mode, V the operating voltage, and T transmission time of one IEEE 802.15.4 beacon with a DIO message in its payload (typically around 3.5 ms for 250 kb/s IEEE 802.15.4 compliant radios).

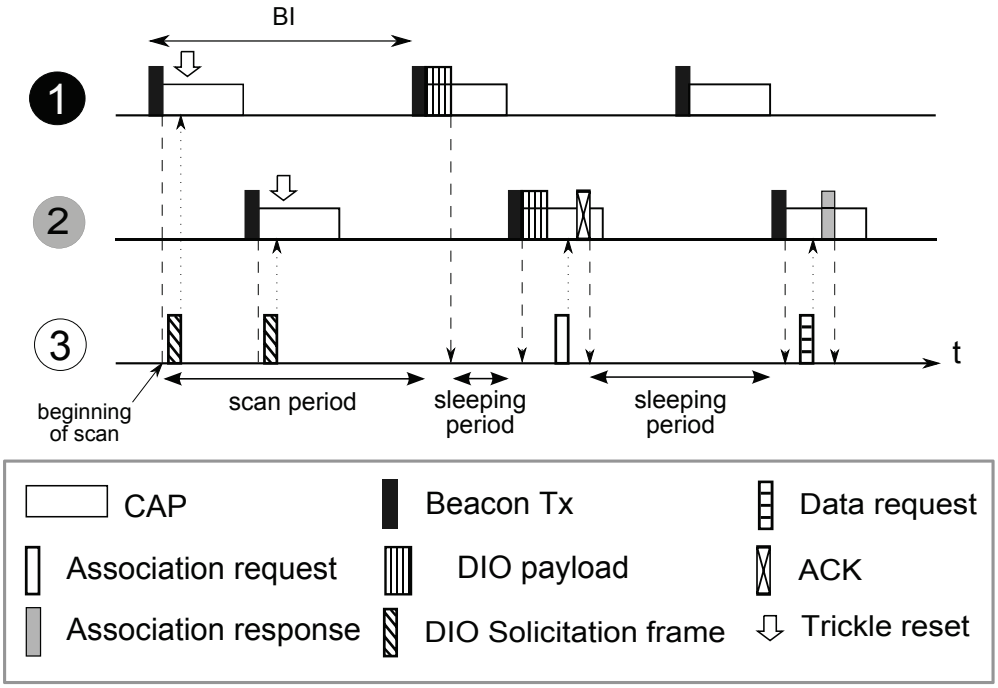


Figure 2.3: Soliciting DIO during the scan period.

Note, however, that in many deployments, BO is not apriori known. In such cases, devices have to scan for longer periods to account for the largest expected BI in presence of multiple PANs [72]. Our scheme in such scenarios introduces no additional delay as long as the preconfigured scan duration is greater than or equal to half the actual BI in the network.

B.3.1 I_{min} Parameter Tuning and Analysis

The successful operation of the proposed scheme requires that, upon solicitation, the subsequent beacon includes a DIO message. To achieve such behavior while keeping the operation of two layers independent, we need to configure the Trickle I_{min} parameter as a function of BI, because the reception of a solicitation frame triggers the Trickle timer reset and the next timer value will be uniformly drawn from the interval $[I_{min}/2, I_{min})$. Thus, to ensure the arrival of the next DIO before the subsequent beacon, the following condition needs to hold:

$$I_{min} \leq \text{BI} - SD, \tag{2.1}$$

where SD denotes CAP duration. Similarly, as previously discussed, the worst case scan period when BO is a priori known, is BI. The optimal performance of Trickle with our scheme is obtained when $I_{min} = \text{BI} - SD$, which ensures the successful operation while having the lowest overhead.

B.3.2 Analysis of DIO Reception Delay

We evaluate here the expected delay of DIO messages encapsulated in periodic beacons. We define the Trickle timer value as random variable X uniformly distributed in $[I/2, I)$, where I is a random variable denoting the current Trickle state. Then, from the link-layer point of view, a DIO message arrives during a beacon interval at instant $X \ mod \ BI$. Delay D is the interval remaining until the transmission of the next beacon:

$$D = BI - (X - \left\lfloor \frac{X}{\text{BI}} \right\rfloor * \text{BI}). \tag{2.2}$$

The expected delay is then:

$$E[D] = \text{BI} - E[X] + E[\left\lfloor \frac{X}{BI} \right\rfloor] * BI. \tag{2.3}$$

Now, recall that I is a discrete random variable in $\{I_{min} \times 2^n\}$, where $n = 0, 1, \ldots, I_{max}$.

We model I with a discrete-time Markov chain shown in Fig. 2.4, where p denotes the probability of the Trickle reset. We can notice from Fig. 2.4 that

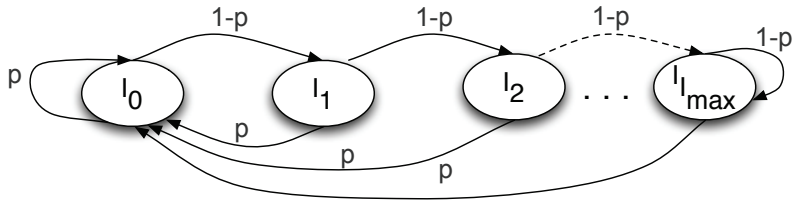


Figure 2.4: Markov chain with I_{max}+1 states for Trickle.

stationary probabilities of states $I_0, \ldots, I_{I_{max}-1}$ follow a geometric distribution with

reset probability p:

$$\Pi_{I_i} = (1 - p)^i p, i = 0, \ldots, I_{max} - 1.$$

The last state, $I_{I_{max}}$ has the stationary probability:

$$\Pi_{I_{I_{max}}} = (1 - p)^{I_{max}}.$$

We can find the expected Trickle timer value as $E[X] = E[E[X|I]]$.

As our scheme uses the *beacon-request* solicitation frame at L2 to reset Trickle, the case $I = I_{min}$ is of a particular interest. From Eq. 2.3, it follows that:

$$E[D|_{I=I_{min}}] = \text{BI} - E[X|_{I=I_{min}}] + E[\left\lfloor \frac{X}{BI} \right\rfloor |_{I=I_{min}}] * \text{BI}. \tag{2.4}$$

Given the condition of Eq. 2.1 and also the fact that the right endpoint is excluded from the uniform interval, term $E[\lfloor \frac{X}{BI} \rfloor]$ goes to zero leaving:

$$E[D \mid I = I_{min}] = \text{BI} - E[X \mid I = I_{min}].$$

Finally, as X is now a uniform random variable in $[I_{min}/2, I_{min})$, the expected DIO delay becomes:

$$E[D|I = I_{min}] = \text{BI} - \frac{3}{4} I_{min}, \; I_{min} \leq \text{BI}. \tag{2.5}$$

We have validated Eq. 2.5 by emulating a real node running the Contiki operating system for constrained devices. We have timestamped the expiration instants of Trickle and the instants of the beacon with DIO transmission. We have configured I_{min} to an approximate value of $BI/2$ (Contiki accepts the values of I_{min} in power of 2). The emulation results over 5000 samples strongly corroborate our analysis with a maximal error of 2.799%.

From Eqs. 2.1 and 2.5, it follows that for setting $I_{min} = \text{BI} - SD$, our scheme introduces the least additional delay to Trickle after reset, while ensuring successful operation.

B.4 Performance Evaluation

To evaluate our scheme, we have used an implementation of the IEEE 802.15.4 *beacon-enabled* mode developed in the context of GreenNet project. To benefit from the Cooja simulator [113] that uses the MSPsim instruction-level emulator of the Tmote Sky platform, our team ported the *beacon-enabled* layer developed for GreenNet motes to the Tmote Sky platform[1]. Note that the only imperfection of Cooja with respect to the real world environment comes from the Unit Disk Graph (UDG) radio channel model. Fig. 2.5 presents the evaluated topology.

Many authors in the literature discussed the method of encapsulating information necessary for topology construction in the beacon payload (parent selection,

[1]Tmote Sky is a commercial clone of TelosB motes discussed in Part I.

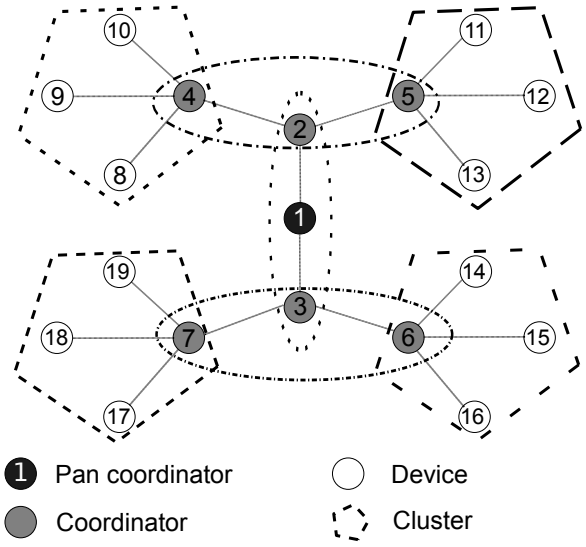


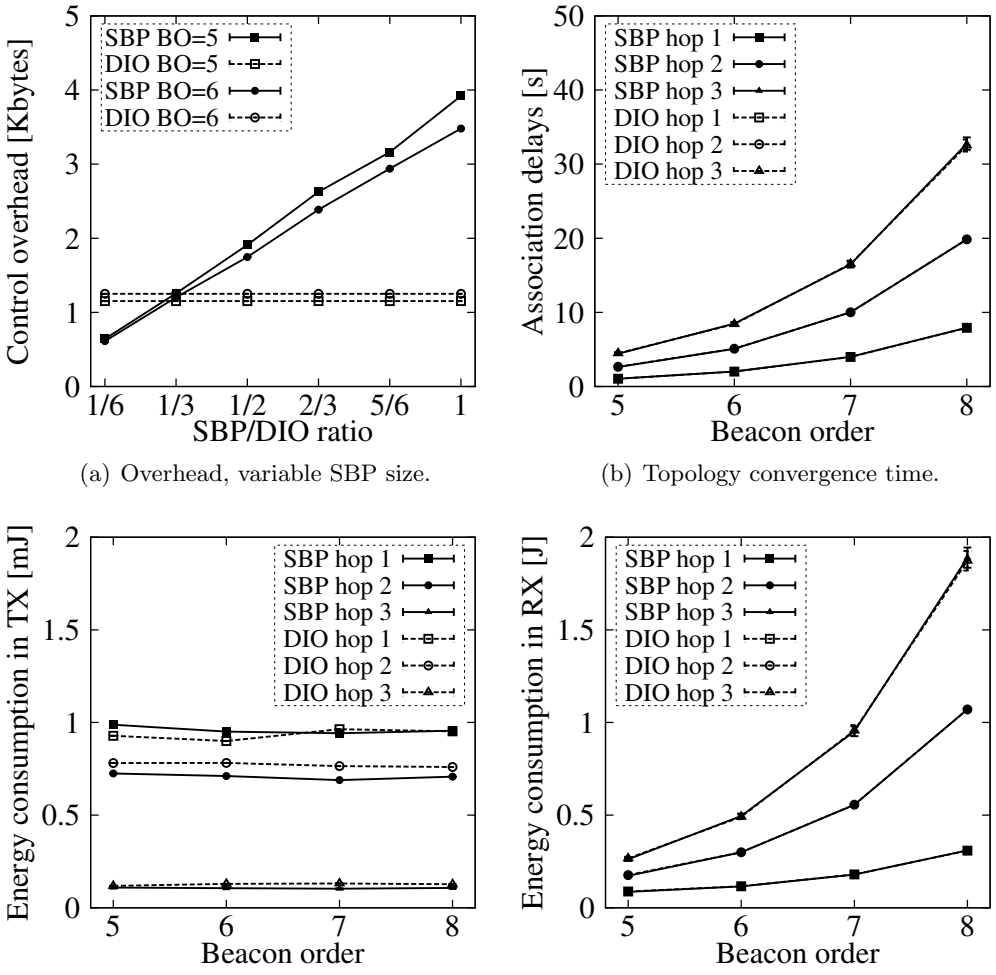
Figure 2.5: Topology used for the evaluation of the proposed scheme.

neighbor discovery) [79, 117, 181]. Consequently, they assume the information to be present in each beacon. As our goal in this chapter was to present benefits in terms of 802.15.4 topology construction, we have compared our scheme against this approach and denote the scheme Systematic Beacon Payload (SBP). To be fair and not to loose the generality of our results, we have studied the effects of varying the SBP message size and how it affects performance. We found that the two schemes have similar performance when the SBP message size is approximately 1/3 of the DIO size (cf. Fig. B.6(a)), that is, when one coordinator from Fig. 2.5 sends 1 DIO message for every 3 beacons with SBP on the average during topology construction. Note that this ratio depends on the duration of the scan period and the configuration of Trickle. For a given implementation, one can easily evaluate such a ratio and derive the gain or loss depending on the message size parameters.

We set the I_{min} Trickle parameter to approximately $BI - SD$ and keep SO equal to 2. We compute the radio energy consumption from the current draw values reported in the Tmote Sky data sheet. We average all the points in the following graphs over 20 emulation runs and show them with 95% confidence intervals.

We can notice in Fig. 2.5 that nodes have only one coordinator in their radio range. We have chosen such topology to focus on topology construction in RPL networks over *beacon-enabled* 802.15.4 and evaluate the effect of our scheme. In this way, we isolate topology construction aspects from the problems related to routing that may depend on the choice of routing metrics or objective functions. Moreover, a single coordinator discovered during the scan period BI means that the solicitation scheme is put under stress. Indeed, if a single DIO message does not arrive with the subsequent beacon upon solicitation, the node will have to initiate another scan period, which would unnecessarily increase the topology convergence delay. Nevertheless, the example topology in Fig. 2.5 is favorable to the proposed

scheme in terms of delay—it does not introduce additional delay in case BO is a priori known, i.e., the first discovered coordinator is also the last one, so a node can initiate the association procedure after the scan period of one BI. However, we discuss the worst case delay in the presence of multiple coordinators in Section B.3.



(a) Overhead, variable SBP size.

(b) Topology convergence time.

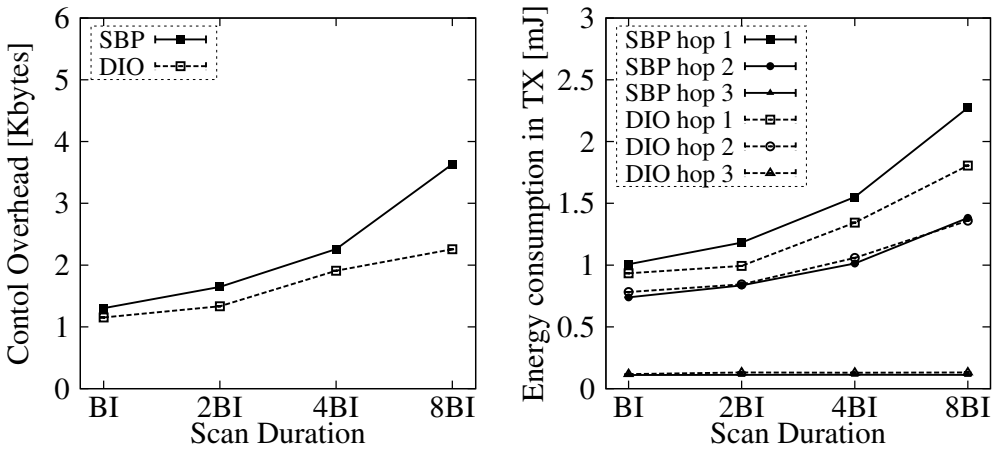
(c) Energy spent in transmission, in mJ.

(d) Energy spent in reception, in J.

Figure 2.6: Results from emulation during topology construction.

Also note that in some cases, the first beacon discovered during the scan period may already contain a DIO message. As the Trickle timer randomly selects its expiration interval and our scheme keeps the operation of two layers independent, it is a lucky outcome. In this case, a node does not need to solicit DIO as detailed in Section B.3. However, a node still has to wait for the expiration of the scan period before initiating its association procedure to ensure that it has discovered all potential coordinators.

We present the results for the case in which two schemes have the most similar performance, i.e., we set the message size of SBP to 1/3 of DIO (cf. Fig.B.6(a)).



(a) Total overhead (b) Energy spent in transmission, in mJ

Figure 2.7: Results from emulation during topology construction for variable scan duration and $BO = 5$.

Larger SBP message sizes result in worse performance while smaller SBP messages result in better performance during topology construction in case BO is a priori known.

B.4.1 Topology Construction

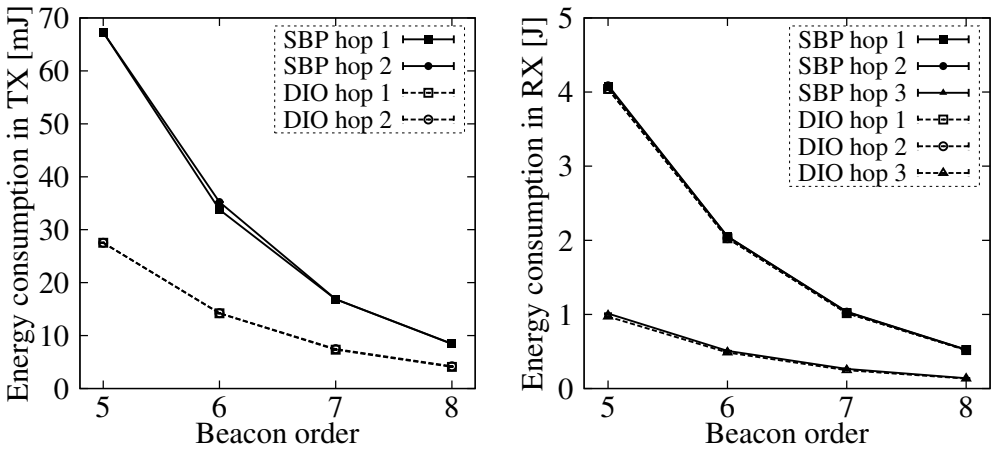
We study the topology construction phase for two cases: 1) BO is a priori known so the scan period can be set to the minimal value of BI; 2) there is no a priori knowledge of BO so nodes use a sub-optimal scan duration to account for the worst case. In both cases, simulations last until the association of the last node.

For case 1), Figs. B.6(b)-B.6(d) present the results for varying BO. We can see in Fig. B.6(b) that our scheme does not introduce any additional delay for the evaluated topology and the results for two schemes are similar within confidence intervals. Fig. B.6(c) shows similar results in terms of cumulative energy spent in transmission, a consequence of the choice of the parameters for two schemes. Notably, coordinators at hop 1 and 2 spend approximately the same energy transmitting beacons. The major part of the energy spent in reception comes from idle listening during the scan period so two schemes perform equally (cf. Fig. B.6(d)).

For case 2), when BO is not a priori known, we vary the scan period. As nodes remain in reception mode much longer, the energy spent in reception makes the major part of the total consumption. Similarly to Figs. B.6(b) and B.6(d), two schemes perform equally. However, as the scan period is longer, there is a larger number of beacons transmitted before the topology converges. We can thus see the effect of the Trickle algorithm and the proposed solicitation scheme (cf. Fig. B.7(a)) that results in energy savings for hop 1 nodes as they transmit beacons the longest until the end of the tree construction (cf. Fig. B.7(b)).

B.4.2 Steady-state

Furthermore, we have evaluated the benefits in terms of energy savings in the steady state, i.e., after topology construction. There was no application traffic in the network and nodes simply duty cycle according to their schedules. The presented results concern 6 minutes of the network operation after the association of the last node. We can see the effect of the reduction in control overhead by the Trickle algorithm in Fig. B.8(a). In particular, FFD nodes (hop 1 and 2) transmit short beacons without any payload most of the time, which results in energy savings both during reception and transmission. During reception, however, a major part of energy consumption comes from active listening during the CAP of each coordinator so this effect is masked (cf. Fig. B.8(b)). Note that in Fig. B.8(a), the consumption of RFDs is zero as there is no application traffic in the network. Also, during the steady state, the reception consumption of FFDs (hop 1 and 2) is the same, as devices remain active during the same amount of time (CAP duration).



(a) Energy spent in transmission, in mJ.

(b) Energy spent in reception, in J.

Figure 2.8: Energy consumption during 6 min of steady state.

B.5 Related Work

The performance of multi-hop IEEE 802.15.4 networks has been well studied during the last years both using probabilistic approaches [106] and simulations [4]. Energy consumption introduced during the scan period is widely recognized as a significant problem. The recent work of Karowski *et al.* [72] lowered this cost by optimizing the number of slots to listen over different channels. Romaniello *et al.* [134] proposed the Multichannel Beacon Train Protocol for faster discovery over multiple channels in the presence of varying beacon intervals. Kohvakka *et al.* discussed a protocol that carries the time offset and the frequency channel in beacons to ease the scanning process for the joining node [79]. It is important to stress that our scheme is agnostic of the scanning process. Namely, the solicitation scheme we propose starts once a

node has discovered all neighboring coordinators.

As the de-facto standard for routing in IP-based WSNs, RPL has been extensively studied in terms of convergence delays, route optimality, path availability, and incurred overheads [172, 75]. Coupled with the common wisdom that cross-layer signaling is necessary for a successful operation of a routing protocol in low power and lossy networks, this fact provides a strong support to the approach presented in our paper.

The work of Pavković *et al.* is closely related to ours [117]. The authors proposed the adaptations to the IEEE 802.15.4 standard to integrate its operation with RPL. Moreover, they proposed an opportunistic version of RPL to improve the delivery of time-sensitive traffic and evaluated the proposal in terms of packet delivery ratio and delay. In recent work [116], they discussed the RPL performance benefits of modifying the IEEE 802.15.4 cluster-tree structure into a "cluster-DAG". Our work was basically motivated by the same problem—the incompatibility of two structures, the IEEE 802.15.4 cluster-tree and the DODAG. While the approach of Pavković *et al.* presents performance improvement, its main drawback is the need for modifications of two standards, RPL and IEEE 802.15.4. We have addressed the same problem from a different perspective—instead of modifying the standards, we provide a means for constructing the RPL DODAG and forming the cluster-tree as its subset. As a consequence, we obtain full compliance with both standards.

B.6 Conclusion

We have presented a scheme that allows coupling *beacon-enabled* IEEE 802.15.4 with the RPL routing protocol. The scheme does not require any modification to both standards. We provide a means for RPL to pass the routing information to the link layer before the IEEE 802.15.4 topology is created by encapsulating RPL DIO messages in beacon frames. The scheme takes advantage of IEEE 802.15.4 command frames to solicit DIO messages. The effect of the command frames is to reset the Trickle timer that governs sending of DIO messages.

We have evaluated the proposed scheme using the Contiki operating system for constrained nodes and the instruction-level Cooja simulator. The results show energy savings during the topology construction phase and in the steady state.

Appendix C

Multiple Redundancy Constants with Trickle

The Trickle algorithm is a timer based control algorithm relying on recursive doubling time intervals and "polite gossip" policy [c_7]. It quickly propagates updates in the network but avoids unnecessary transmissions. Due to its wide-spread use, it has been standardized in RFC 6206 [90]. Apart from IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [178] that utilizes Trickle for topology maintenance, protocols such as Multicast Protocol for Low Power and Lossy Networks [59] and other proposals [183, 45, 58] build upon it, leveraging Trickle's benefits. This makes the understanding of its behavior crucial for performance optimization of control overhead. The related work in the literature [12, 77, 167, 104, 5, 76] has already tackled many aspects of its operation through analytical models and performance studies. Still, all authors seem to consider the crucial "politeness" parameter – the redundancy constant, controlling the number of redundant transmissions in the network – fixed and common for all nodes in the network. To the best of our knowledge, this is also true for real world deployments of Trickle-based networks.

In this chapter, we model and study the operation of Trickle, which leads to a better understanding of the impact of its redundancy constant. We demonstrate that the usage of a common redundancy constant for the whole network leads to communication unfairness when the underlying topology density is not homogeneous. The root cause of this unfairness is the increased probability of transmission of nodes with less neighbors in their radio vicinity. This results in uneven transmission load, e.g. message count, across the network. Moreover, in battery powered networks, these nodes with a higher transmission probability will cease functioning sooner because of on board energy depletion as broadcasting is very expensive in Wireless Sensor Networks (WSNs). We model the individual transmission probabilities with individual node redundancy constants across the network. The model with multiple redundancy constants can be numerically resolved for arbitrary topologies but also simplified to closed-form in specific cases.

From the model's results, we propose a simple heuristic algorithm in order to improve Trickle fairness by a local computation of each redundancy constant as a function of the number of neighbors. We demonstrate the resulting improvements in terms of transmission load balance both by leveraging our analytical model results and by emulating constrained-node networks running the full Contiki Operating System network stack.

The main contributions of this chapter are the following:

- A new probabilistic model estimating the message count and average transmission probabilities of individual nodes in steady state networks. This model works for arbitrary topologies without any synchronization requirement, and accounts for multiple redundancy constants among nodes,
- A demonstration of transmission load unfairness in networks utilizing a fixed redundancy constant among nodes,
- A new algorithm improving fairness in the network by locally computing the redundancy constants as a function of the number of neighbors in the node's radio vicinity,
- A validation of the model and an evaluation of the proposed algorithm improvements. The emulation uses highly accurate instruction-level execution of the binary file that contains our code and the Contiki network stack and normally runs on real hardware.

The reader is expected to be familiar with the Trickle algorithm, as presented in Appendix A. We use the same notation throughout this chapter and model Trickle networks in steady state, such that $I = I_{min}\ 2^{I_{max}}$, and focus on the effect of the redundancy constant.

The rest of the chapter is structured as follows. In Section C.1, we discuss in details the related work regarding the Trickle algorithm. Section C.2 presents our probabilistic model design. We validate the model and discuss common redundancy constant unfairness in Section C.3. In Section C.4, we present our heuristic algorithm that locally computes each redundancy constant and discuss its achieved improvements. Finally, we conclude and discuss future work in Section C.5.

C.1 Related Work

Due to its wide-spread use, the Trickle algorithm has been subject to many studies [12, 77, 167, 104, 5, 76]. Becker *et al.* [12] develop a model to study the propagation time of new information in a network using Laplace transforms.

Meyfroyt *et al.* [104] recently published a model generalizing the algorithm by introducing the *listen-only* parameter η. In the standardized version of Trickle [90] and the original paper [91] $\eta = \frac{1}{2}$ and is introduced in order to avoid broadcast storms in unsynchronized networks at the beginning of intervals, by forcing nodes to keep listening before attempting transmissions (i.e. listen-only period). The authors demonstrate that using a short listen-only period provides advantage in terms of smaller propagation time, but in the same time increases the number of transmitted messages in the network. They derive the cumulative distribution function of inter-transmission times for large number of nodes in a steady state, unsynchronized, single cell network (i.e. all nodes are within each other's radio range and $I = I_{min}\ 2^{I_{max}}$) and the mean number of transmissions (message count). Their model is based on the observation that the process of nodes attempting to broadcast

behaves as a Poisson process as the number of nodes in the single cell network grows large, under the assumption of uniformly distributed interval skew in an unsynchronized network. The model provides very accurate estimates but is limited by the single-cell network assumption. The authors briefly discuss the extension of a single-cell model to multi-cells by introducing a fixed radio range for each node in a grid topology. Apart from being limited by the topology assumption, this approach requires knowledge of nodes' transmission ranges which is not realistic in wireless deployments. Moreover, the approach does not allow insights on individual node behavior. It rather considers the behavior of the network as a whole. Nevertheless, the model gives a very useful, global view study of the effect of Trickle parameters on the message count.

On the opposite, Kermajani *et al.* [77] approach the problem of estimating the Trickle message count in steady state by deriving the average probability P that a node in the network will transmit in a given interval. Then, the average message count in a given interval is simply $N \times P$, where N denotes the number of nodes in the network. To derive P, the authors make two assumptions: i) a uniformly random spatial distribution of nodes, ii) synchronized Trickle intervals (i.e. the interval start time is the same for all nodes). The first assumption is used in order to derive the probability of two nodes being each other's neighbors, assuming a common, known, radio coverage range of each node in the network. Regarding the second assumption on synchronized intervals, which is rarely met in practice, the authors demonstrate and discuss surprisingly small differences of their model with simulation results without synchronization. In respect to the model of Meyfroyt *et al.* [104], the approach of Kermajani *et al.* [77] implicitly supports multi-cell topologies.

Other authors [167, 5, 76] have studied Trickle and its performance in the specific use case of RPL and how it affects the convergence and route optimality of the Destination Oriented Directed Acyclic Graph (DODAG) building process. Vallati and Mingozzi [167] observe that the setting of the redundancy constant in RPL highly affects the constructed route optimality. They demonstrate that lower values of the redundancy constant decrease energy consumption but also decrease the quality of constructed routes, due to the fact that some nodes stay silent. This phenomena is related to the fact that Trickle aims at providing equal transmission probability in the long run, while for routing purposes it is important that every node shares its state on the shorted possible scale [167]. The authors tackle the problem by reducing the listen-only period with every suppressed transmission. Their approach results in better short-term fairness but at the cost of increased number of transmitted messages [104].

A common point on published Trickle models and optimizations [12, 104, 77, 167] (and deployments) is that they all consider a common, fixed redundancy constant among nodes. The common redundancy constant leads to *long-term* unfairness as nodes with less neighbors have less incoming packets and thus a higher probability to transmit, and will therefore deplete their available energy source sooner. In order to model this phenomena, like Kermajani *et al.* [77] we calculate the message

count in steady state using the average transmission probability. Instead of making assumptions on topology, we assume networks where the number of radio neighbors of a node is a locally available information, which is often met in practice. We then consider that each node may have a different redundancy constant and calculate the average probability of transmission *per node*, which allows us to estimate the average message count individually, and for the whole network. This sets ground for the testing of different algorithms for the computation of the redundancy constant. We consequently give a heuristic algorithm and demonstrate its effects on long-term load distribution. In our model, we also relax the assumption on synchronized intervals and consider a uniformly distributed interval skew among nodes, which is often met in practice.

C.2 Modelling the Trickle Message Count

Our model calculates the message count of individual nodes in a steady state Trickle-based network. Similarly to Kermajani *et al.* [77], we derive the average probability of transmission. However, we use a different decomposition that allows us to extend their approach in order to calculate per node probabilities, rather than the network average. Hence we can give insights on the fairness of the algorithm. To render the model more practical, we do not make explicit assumptions on topology. Rather, we assume that each node i is able to know its number of neighbors y_i. Therefore, our model requires for each node its redundancy constant K_i, and its neighbors list.

The main idea of our analysis is to express the average probability of transmission of a node as a function of the transmission probabilities of its neighbors. This will yield a system of N equations with N unknowns, where N is the number of nodes in the network, that can be numerically resolved.

Distribution of Transmission Times. In networks with synchronized Trickle intervals, transmission times simply follow a uniform distribution [77]. However, with the lack of synchronization, this does not longer hold. Let $X_1, \ldots, X_{y_i}$ be i.i.d. random variables of uniform distribution modeling the transmission time positions of the y_i nodes into an interval of length I. Let T be the selected transmission time of node i, $T \in [\frac{1}{2}\mathrm{I}, \mathrm{I}]$. Let Y_T denote the number of selected transmission times before T. Let n be the positive integer that denotes the position of transmission time of node i in the set of increasingly ordered transmission times. The probability that n is selected by node i and by its neighbors is equal to $P(Y_T = n-1)$. Y_T can be shown to follow a binomial distribution with parameters y_i and $\frac{T}{I}$ [26].

C.2.1 Probabilistic Model

The average probability that node i will send a message in a given interval is denoted $P_{TX}[i]$. A node will surely transmit in the case where its number of neighbors y_i is less than its redundancy constant K_i, because the counter c can never reach K_i. Otherwise, the node transmits in two cases: i) if it selects any of the first K_i

transmission times, ii) if it selects any of the last $y_i + 1 - K_i$ transmission times and at most $K_i - 1$ of its neighbors have already transmitted. Consequently, $P_{TX}[i]$, can be written as follows:

$$P_{TX}[i] = \begin{cases} 1, & y_i < K_i \\ P_F[i] + P_{LO}[i], & y_i \geq K_i. \end{cases}$$

where:

- $P_F[i]$ is the probability that node i selects one of the first K_i transmission times. We can find this probability simply as: $P_F[i] = \sum_{n=0}^{K_i-1} P(Y_T = n)$.
- $P_{LO}[i]$ is the probability that node i selects any of the last $y_i + 1 - K_i$ transmission times and at most $K_i - 1$ nodes, with a lower transmission time than node i, will transmit before it. We refer to this probability as the last opportunity transmission probability. This probability depends on $P_{TX}[j]$, where j is a neighbor of node i.

Fig. 3.1 presents an overall summary of the probability decomposition. The idea is to decompose the algorithm in outcomes and to compute the probabilities associated to each of them. Then, each sub-problem is tractable and the model can lead to numerical computations in the general case, and to closed form solutions for specific topologies.

Last Opportunity Transmission Probability. We are considering the case where node i selected one of the last $y_i + 1 - K_i$ transmission times. The probability that at most $K_i - 1$ nodes, with a lower transmission time than node i transmit before, depends on transmission time of node i. We will compute the probability $P_{LO}[i]$ by conditioning on transmission time of node i. As $Y_T \in \{K_i, \ldots, y_i\}$, $P_{LO}[i]$ can be derived as:

$$P_{LO}[i] = \sum_{n=K_i}^{y_i} P_{LO}[i \mid Y_T = n] \times P(Y_T = n). \tag{3.1}$$

Let B_{set} be the set of n neighbors of node i, denoted by $\{1, \ldots, n\}$ whose transmission times are lower than the one of node i. Let $\Re$ be the set composed of $\binom{y_i}{n}$ possible sets of nodes B that possibly match B_{set}.
Therefore, $P_{LO}[i \mid Y_T = n]$ can be obtained as:

$$P_{LO}[i|Y_T = n] = \frac{1}{\binom{y_i}{n}} \sum_{B \in \Re} P_{LO}[i|Y_T = n \wedge B_{set} = B]. \tag{3.2}$$

The probability that node i transmits in this case, $P_{LO}[i \mid Y_T = n \wedge B_{set} = B]$, is the probability that at most $K_i - 1$ nodes of B transmit before:

$$P_{LO}[i \mid Y_T = n \wedge B_{set} = B] = \sum_{j=0}^{K_i-1} \gamma(j, n, B), \tag{3.3}$$

where $\gamma(j, n, B)$ denotes the probability that j nodes of the set $B = \{1, \ldots, n\}$ transmit before node i. By definition of P_{TX}, we have:

$$\gamma(0, n, B) = \prod_{l=1}^{n}(1 - P_{TX}[l]),$$

$$\gamma(1, n, B) = \sum_{i_1=1}^{n}[P_{TX}[i_1] \times \prod_{\substack{l=1 \\ l\neq i_1}}^{n}(1 - P_{TX}[l])],$$

$$\gamma(2, n, B) = \sum_{i_1=1}^{n}\sum_{\substack{i_2=1 \\ i_2\neq i_1}}^{n}[P_{TX}[i_1] \times P_{TX}[i_2] \times \prod_{\substack{l=1 \\ l\neq i_1, i_2}}^{n}(1 - P_{TX}[l])].$$

Then:

$$\gamma(k, n, B) = \sum_{i_1=1}^{n}\sum_{\substack{i_2=1 \\ i_2\neq i_1}}^{n}\ldots\sum_{\substack{i_k=1, \\ i_k\neq i_1,\ldots,i_{k-1}}}^{n}[P_{TX}[i_1] \times P_{TX}[i_k] \times \prod_{\substack{l=1, \\ l\neq i_1,\ldots,i_k}}^{n}(1 - P_{TX}[l])].$$

But we can express $\sum_{i_1=1}^{n}\ldots\sum_{\substack{i_k=1, \\ i_k\neq i_1,\ldots,i_{k-1}}}^{n}$ as $\sum_{(i_1,\ldots,i_k)\in A}$, where A is a set of combinations without repetition of k elements selected among n. Thus:

$$\gamma(k, n, B) = \sum_{(i_1,\ldots,i_k)\in A}[P_{TX}[i_1] \times \ldots \times P_{TX}[i_k] \times \prod_{\substack{l=1, \\ l\neq i_1,\ldots,i_k}}^{n}(1 - P_{TX}[l])].$$

The right side can be rearranged in:

$$\sum_{v=(i_1,\ldots,i_k)\in A}[\prod_{m=1}^{k}P_{TX}[v[m]] \times \prod_{\substack{l=1, \\ l\neq i_1,\ldots,i_k}}^{n}(1 - P_{TX}[l])],$$

which leads to:

$$P_{LO}[i] = \sum_{n=K_i}^{y_i}P(Y_T = n) \times \frac{1}{\binom{y_i}{n}}\sum_{B\in\Re}\sum_{j=0}^{K_i-1}\gamma(j, n, B). \tag{3.4}$$

The main steps of our derivation can be followed in Fig. 3.1: (Probability that node i selects the $(n+1)^{th}$ transmission time) AND (Probability to have the set B of n neighbors with lower transmission time than node i) AND (Probability that at most $K_i - 1$ nodes in set B transmit). For example, let us consider the case where node i has $y_i = 4$ neighbors, $\{a, b, c, d\}$, and $n = 2$ of them have selected a transmission time lower than its own. We can have $\binom{4}{2}$ different sets of neighbors

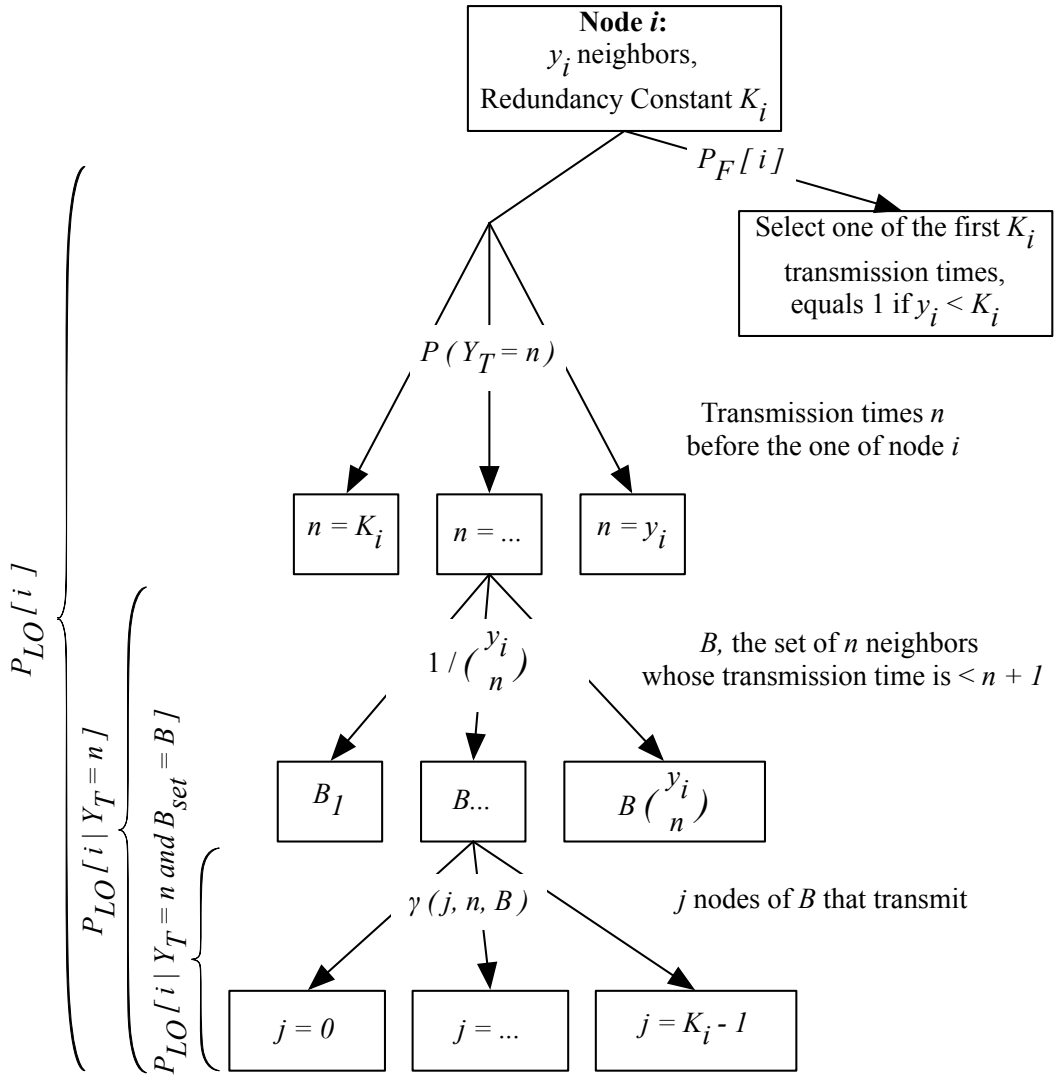


Figure 3.1: Probability decomposition of the model. We present possible events in boxes and their associated probabilities on the corresponding arrows.

$B_1 = \{a, b\}, B_2 = \{a, c\}, B_3 = \{a, d\}, B_4 = \{b, c\}, B_5 = \{b, d\}, B_6 = \{c, d\}$, whose nodes have a lower transmission time. For instance, when $B = B_1 = \{a, b\}$ and $K_i = 2$, node i will transmit either if $j = 0$ nodes of B transmit, or if $j = 1 = K_i - 1$ nodes of B transmit. The probability that j nodes of the set B transmit is $\gamma(j, n, B)$.

We now have $P_F[i]$ and $P_{LO}[i]$ expressed according $P_{TX}[j]$, where j is a neighbor of node i. In order to find $P_{TX}[i]\ \forall i \in \{1, \ldots, N\}$, following our decomposition, we need to solve the N equations with N unknowns, i.e. the system $P_{TX}[i] = P_F[i] + P_{LO}[i]$. The solutions of the system are the average probabilities of transmission for each node in the network and in the same time the average message count per node during one interval. In its general form, the system of equations models arbitrary network topologies and can be resolved numerically. For specific topologies which are outside of the scope of our work, a closed form solution may be obtained.

Due to its complexity, the general form of the model does not allow a direct practical implementation for constrained devices. However, its numerical resolution gives precise insights on node behavior and where the imbalance in the network occurs. Based on this, in Section C.4 we propose a practical, heuristic approach that is easily computable locally.

C.3 Model Validation and Trickle Unfairness

To validate our model we implemented a tool resolving the model in Python and Sage [1], an open-source computational software program. We emulate the Tmote Sky sensor motes running Contiki Operating System, by using the MSPSim emulator, and the Cooja simulator, in order to obtain real world results of Trickle. We use the Trickle application-level library code available in Contiki. The same binary file used for emulations runs on real hardware without any modifications. Note that due to the use of the emulator, some imperfections of results in respect to the real deployments come from the Unit Disk Graph (UDG) radio model in Cooja. We validate the model for 49 node networks: i) using a 7×7 grid topology, to demonstrate unfairness, ii) using a randomly generated topology, to demonstrate the validity of the model for the general case. Transmission range for the grid topology was $R = \sqrt{2}$, with the average node degree of 6.37. In the case of randomly generated topology, the average node degree was 3.92. We average emulation results over 30 runs, and calculate the model for the same topology based on the list of neighbors of each node. We count the number of transmissions of each node over 10 steady state Trickle intervals of 16 seconds. We calculated 95% confidence intervals but do not present them on the graphs for the sake of clarity, as they are graphically indistinguishable from the plotted averages.

Fig. 3.2 presents the results for the total message count in the network using a common, fixed redundancy constant among nodes. The model accurately predicts the number of messages in the network. Numerical values of maximum, minimum probabilities, variance and their comparison with the emulation results are shown

[1] http://sagemath.org

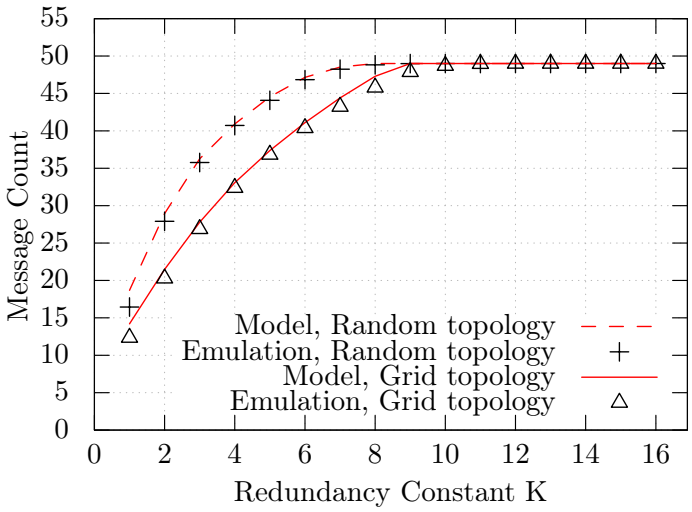


Figure 3.2: Message count in networks with fixed redundancy constant.

in Tables 3.1 and 3.2. The imperfections of the model come from the fact that with the lack of synchronization of Trickle intervals among nodes, if we consider a node with y neighbors, y is only the mean number of transmission times that can occur during one interval. Also, the assumption that transmission probabilities of nodes are independent events does not hold. As discussed by Kermajani *et al.* [77], a transmission performed by a node causes the increment of the counter c and therefore decreases the transmission probability of its neighbors. Nevertheless, emulation results are obtained by running the binary file that normally executes on real hardware. They show that our model provides accurate estimations of the average transmission probabilities of nodes in the network, and consequently of the message count.

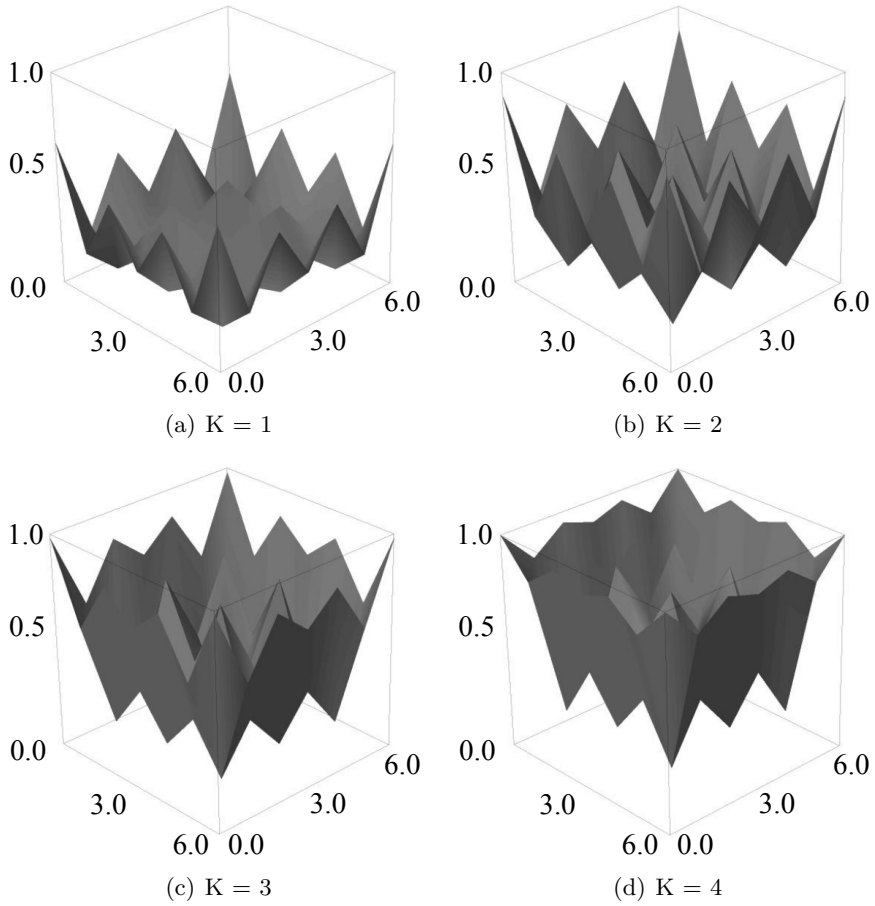
Trickle Unfairness with Fixed Redundancy Constant. As most real world deployments using Trickle utilize a common, fixed redundancy constant among nodes, we demonstrate the transmission unfairness that arises due to the heterogeneous network topologies, as nodes do not have the same number of neighbors. Fig. 3.3 presents the three dimensional graphs on probability of transmission calculated by our model for the grid topology, where the effects are easily noted. We present results for $K \in \{1, 2, 3, 4\}$, as the probabilities quickly approach 1.0 for larger K (average node degree of 6.37). Inside the grid with 8 neighbors ($R = \sqrt{2}$), for $K = 1$, we can see that the nodes have average transmission probabilities of approximately 0.2, while the nodes on the edges with 5 neighbors have around 0.5, and the nodes in the corners with 3 neighbors on the average transmit with the probability of approximately 0.7. Increasing the redundancy constant increases the transmission probabilities in the network. However, as the number of neighbors can be considered fixed, nodes with less neighbors are affected more and their transmission probabilities increase faster in respect to those in the middle of the grid. This can be best seen for the case $K = 4$ in Fig. 3.3(d), as the nodes in the corners of

Table 3.1: Model and emulation results on 7×7 grid, $R = \sqrt{2}$, for $K \in \{1, 2, 3, 4, 5, 6\}$

results / redundancy constant	model $K = 1$	emul. $K = 1$	model $K = 2$	emul. $K = 2$	model $K = 3$	emul. $K = 3$	model $K = 4$	emul. $K = 4$	model $K = 5$	emul. $K = 5$	model $K = 6$	emul. $K = 6$
max probability	0.673	0.606	0.887	0.896	0.980	0.983	0.999	1.0	0.999	1.0	0.999	1.0
min probability	0.070	0.05	0.084	0.05	0.116	0.153	0.173	0.22	0.295	0.38	0.501	0.493
variance	0.03217	0.02466	0.06402	0.05030	0.08261	0.05736	0.08553	0.06077	0.06401	0.05158	0.03268	0.03339

Table 3.2: Model and emulation results on 49 node random topology and 3.92 average node degree, for $K \in \{1, 2, 3, 4, 5, 6\}$

results / redundancy constant	model $K = 1$	emul. $K = 1$	model $K = 2$	emul. $K = 2$	model $K = 3$	emul. $K = 3$	model $K = 4$	emul. $K = 4$	model $K = 5$	emul. $K = 5$	model $K = 6$	emul. $K = 6$
max probability	0.853	0.773	0.999	0.996	0.999	1.0	0.999	1.0	0.999	1.0	0.999	1.0
min probability	0.070	0.056	0.033	0.083	0.095	0.146	0.233	0.28	0.427	0.396	0.663	0.623
variance	0.04783	0.03346	0.08453	0.07687	0.08518	0.07773	0.05276	0.04963	0.0237	0.02392	0.00772	0.00750



(a) K = 1 (b) K = 2

(c) K = 3 (d) K = 4

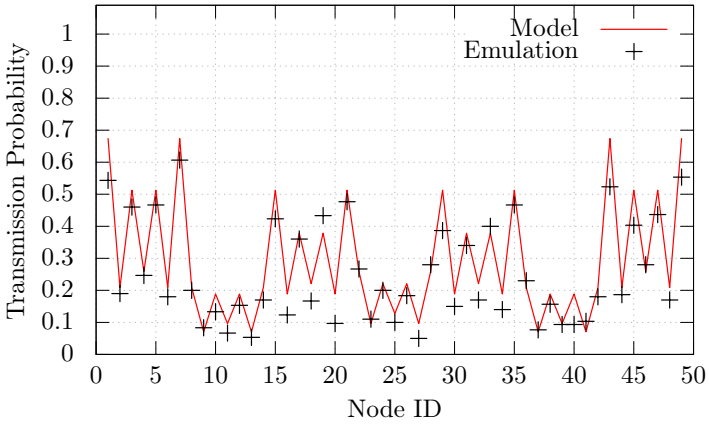
Figure 3.3: Transmission probability of nodes in the grid estimated by the model for fixed redundancy constant.

the grid transmit with probability 1.0, while nodes on the edges have probability 0.85, and nodes inside the grid have probability 0.45.

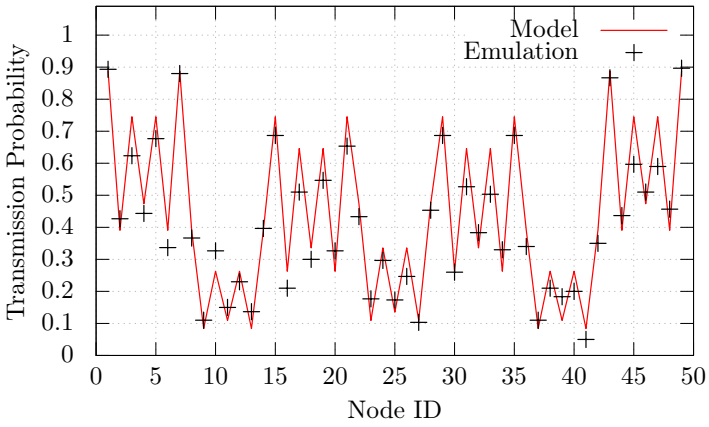
To validate the estimations of our model, we have confronted the results with emulations. For the sake of brevity, Figs. 3.4 and 3.5 presents the comparison for $K \in \{1, 2, 3\}$, in the grid and the randomly generated topology.

C.4 Local Computation of the Redundancy Constant to Improve Fairness

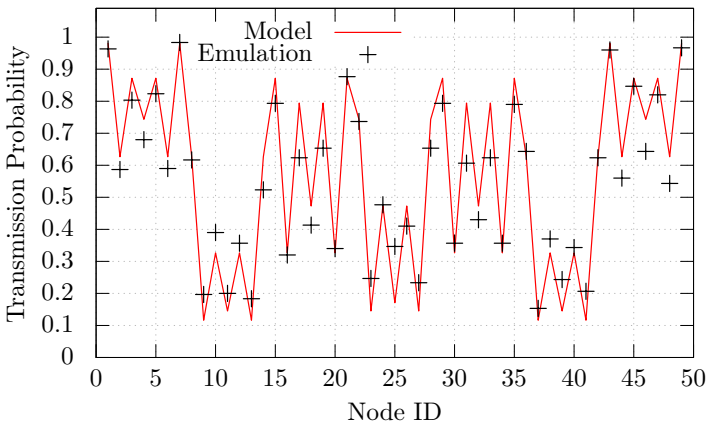
As discussed, the average transmission probability of a node in the network depends on the number of neighbors and the redundancy constant. The usage of a fixed redundancy constant in the network causes unbalanced transmission load and may cause early depletion of energy sources of nodes with less neighbors. Notice that the number of neighbors is generally available locally due to the common use of either Neighbor Advertisement/Solicitation control packets or L2 synchronization



(a) Grid, K = 1

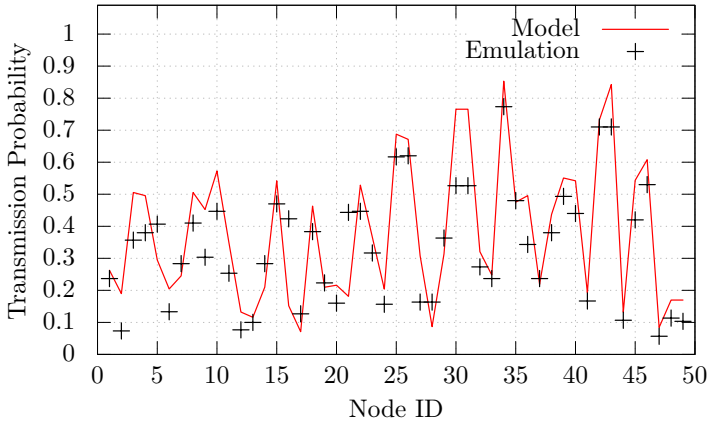


(b) Grid, K = 2

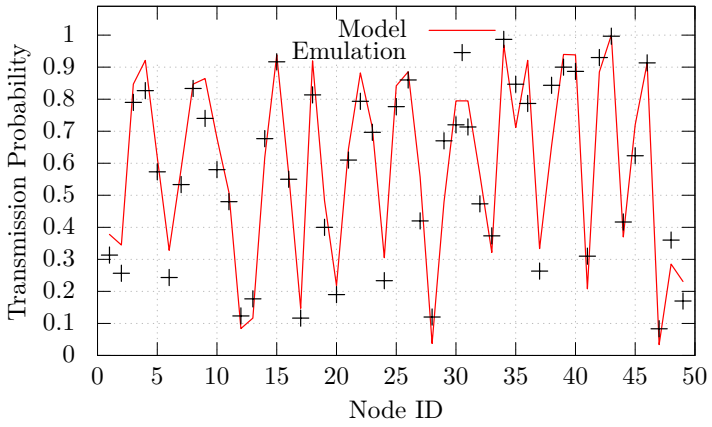


(c) Grid, K = 3

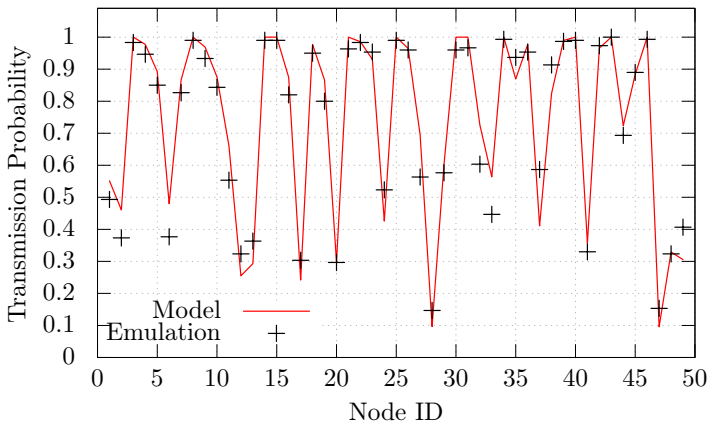
Figure 3.4: Transmission probability of nodes in the grid topology.



(a) Random, K = 1



(b) Random, K = 2



(c) Random, K = 3

Figure 3.5: Transmission probability of nodes in a random topology.

mechanisms. We leverage this fact to introduce multiple redundancy constants among nodes in the network, dependent on the number of neighbors.

Although our first attempt was to derive a closed form expression that will provide a locally optimal value of K_i, due to the complexity of the model this remains an open problem. Instead, we propose a simple calculation of K_i feasible on constrained devices. The idea is to increment K_i for each redundancy `step` number of neighbors. On the other hand, parameter redundancy `offset`, specifies the number of neighbors for each node that corresponds to the minimal value of $K = 1$. The calculation is outlined in Algorithm 9.

Algorithm 9 Local calculation of the redundancy constant K_i

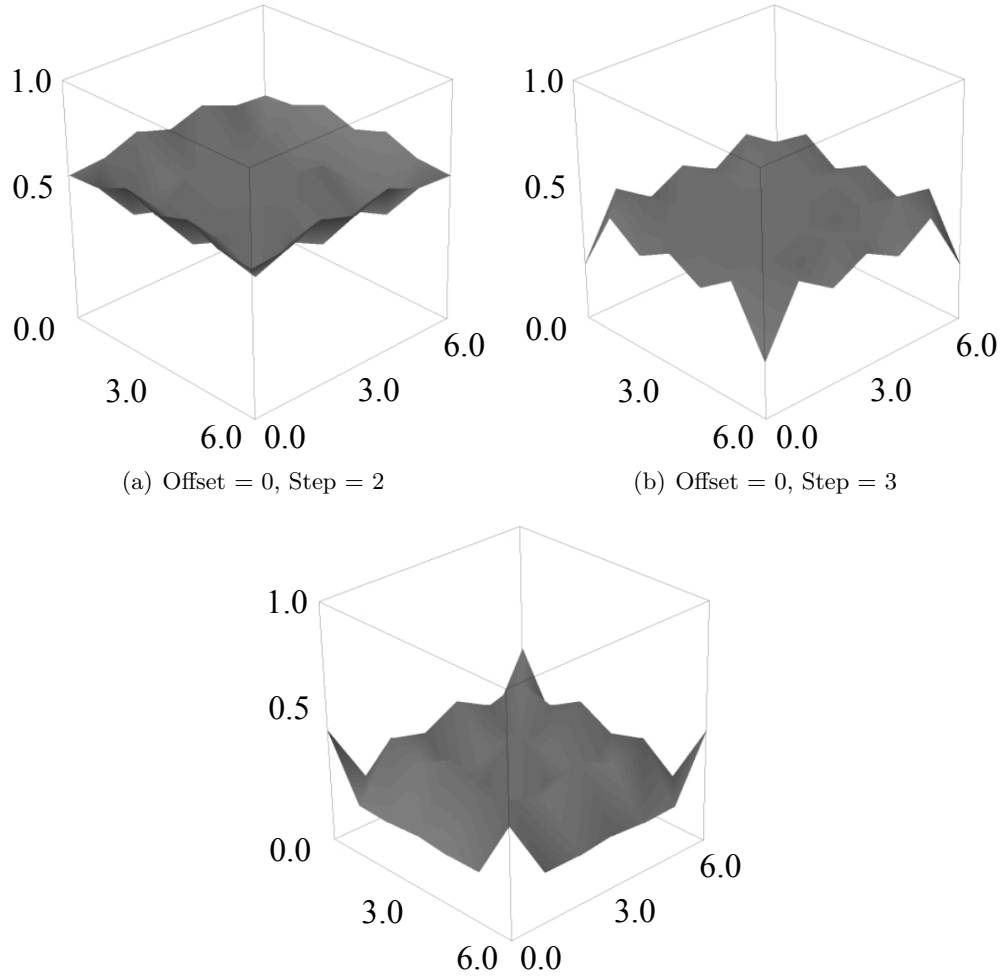
1: **procedure** CALCULATE_K($num\_neighbors, step, offset$)
2: **if** num_neighbors $\leq$ offset **then return** 1
3: **else**
4: **return** $\lceil \frac{num\_neighbors - offset}{step} \rceil$

We show the effect estimated by the model of the locally computed redundancy constant for the most interesting combinations of parameters in Figs. 3.6 and 3.7. We also confront the estimations with emulation results in Table 3.3.

We can see that the use of the locally computed redundancy constant greatly reduced the effects observed in Fig. 3.3. Depending on the parameters passed to the procedure, we note that the effect can be either reversed such that the nodes in the corners transmit with smaller probability than the nodes inside the grid, or reduced which is the case for nodes on the edges of the grid. Clearly, the absolute value of the ideally balanced transmission probability in the network depends on the requirements of the application actually using Trickle.

As an example of better load distribution with our algorithm, we can consider a grid network, as above, where the nodes can, for instance, send 1000 messages before depleting their battery. Consider the maximum transmission probability for $K = 1$ in Table 3.1 (0.673 / 0.606). This will force the first node to shut down after 1480/1650 steady state Trickle intervals, whereas in a network using $K \in \{1, 2\}$, the maximum probability will be 0.479/0.493 (cf. Table 3.3) and will force the first node to shut down only after 2087/2028 intervals.

Parameter Selection. In original Trickle, the redundancy constant K is a parameter that effectively depends on the application requirements. With our algorithm, we extend this concept in order to catch the topology characteristics and to provide a better transmission load distribution among nodes. However, both redundancy `step` and redundancy `offset` effectively depend on the application using Trickle and are semantically equivalent to K. The notion of "redundancy" from the application point of view is in our case defined as a function of the network topology, i.e. how many transmissions are needed for a given neighborhood to reach application needs. For instance, with `step` $= 2$ and `offset` $= 0$ application specifies that a transmission should be suppressed when at least half of the neighbors have advertised their state as consistent. In parallel, `step` regulates the granularity of

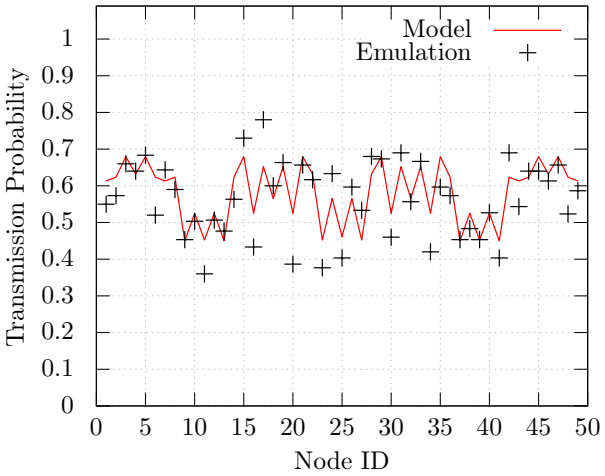


(a) Offset = 0, Step = 2

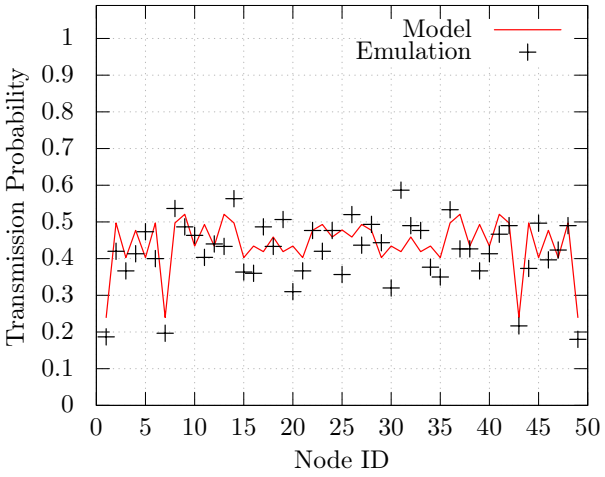
(b) Offset = 0, Step = 3

(c) Offset = 2, Step = 3

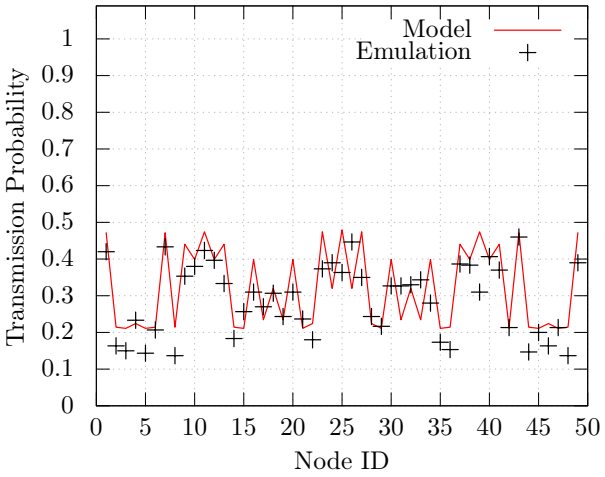
Figure 3.6: Transmission probability of nodes in the grid estimated by the model for locally-computed redundancy constant.



(a) Offset = 0, Step = 2



(b) Offset = 0, Step = 3



(c) Offset = 2, Step = 3

Figure 3.7: Comparison of model estimations with emulation results for locally-computed redundancy constant.

local K_i increments This directly affects the distribution of the transmission load in the network. Thus, instead of blindly defining K, the application will have a finer control on the redundancy depending on the topology. In the same time it achieves better a transmission load distribution.

Table 3.3: Model and emulation results for the locally computed redundancy constant on the grid. To obtain $K \in \{1, 2\}$, we used $offset = 2$, $step = 3$, and for $K \in \{1, 2, 3\}$, $offset = 0$, $step = 3$.

results / redundancy constant	model $K \in \{1, 2\}$	emulation $K \in \{1, 2\}$	model $K \in \{1, 2, 3\}$	emulation $K \in \{1, 2, 3\}$
average message count	15.734	15.326	21.587	21.66
max probability	0.479	0.493	0.520	0.586
min probability	0.011	0.15	0.239	0.213
Variance	0.01188	0.00947	0.00511	0.00800

C.5 Conclusion

In this chapter we presented a model of the Trickle algorithm that estimates the message count in steady state. We do this by calculating average transmission probabilities of individual nodes in the network. This allowed us to demonstrate load misbalance and unfairness of the algorithm when used with a common redundancy constant in the network. The root cause of the unfairness is the heterogeneity of the underlying topology as nodes do not have the same number of radio neighbors in their range. As a consequence, with a common redundancy constant, nodes with less neighbors transmit Trickle broadcast messages more often. We validated our model by comparing it with emulation results and demonstrated its high accuracy. In order to improve the fairness of Trickle, we proposed a simple heuristic algorithm that locally computes the redundancy constant of a node based on the number of neighbors in its vicinity. We demonstrated that by using our algorithm, nodes in the network achieved better transmission load distribution. However, deriving an optimal value of the redundancy constant that will perfectly balance the transmission probabilities of nodes in heterogeneous topologies remains an open problem that we plan to study as future work.

Bibliography

[1] Nadeem Ahmed, Salil S. Kanhere, and Sanjay Jha. The holes problem in wireless sensor networks: A survey. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(2):4–18, April 2005. 58

[2] Olivier Alphand, Etienne Duble, Andrzej Duda, Michel Favre, Roberto Guizzetti, Martin Heusse, and Franck Rousseau. GreenNet: a Wireless Sensor Network for Harvested Nodes. Technical report, Grenoble Informatics Laboratory, 2012. 142

[3] Diego Altolini, Vishwas Lakkundi, Nicola Bui, Cristiano Tapparello, and Mattia Rossi. Low Power Link Layer Security for IoT: Implementation and Performance Analysis. In *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 919–925, July 2013. 60, 61

[4] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Vincenzo Neri. Reliability and energy efficiency in multi-hop IEEE 802.15.4/ZigBee Wireless Sensor Networks. In *Computers and Communication (ISCC), 2010 IEEE Symposium on*, pages 336–341. IEEE, 2010. 150

[5] Emilio Ancillotti, Raffaele Bruno, Marco Conti, Enzo Mingozzi, and Carlo Vallati. Trickle-l2: Lightweight link quality estimation through trickle in rpl networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, pages 1–9, June 2014. 153, 154, 155

[6] Idris M. Atakli, Hongbing Hu, Yu Chen, Wei Shinn Ku, and Zhou Su. Malicious node detection in wireless sensor networks using weighted trust evaluation. In *Proceedings of the 2008 Spring Simulation Multiconference*, SpringSim '08, pages 836–843, San Diego, CA, USA, 2008. Society for Computer Simulation International. 56

[7] Atmel. AT86RF231. `http://www.atmel.com/images/doc8111.pdf`. 10, 41

[8] Atmel. ATmega128L. `http://www.atmel.com/images/doc2467.pdf`. 8

[9] Marco Avvenuti, Paolo Corsini, Paolo Masci, and Alessio Vecchio. Increasing the efficiency of preamble sampling protocols for wireless sensor networks. In *Mobile Computing and Wireless Communication International Conference, 2006. MCWC 2006. Proceedings of the First*, pages 117–122, Sept 2006. 19

[10] Baruch Awerbuch, Reza Curtmola, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. Mitigating byzantine attacks in ad hoc wireless networks. *Department of Computer Science, Johns Hopkins University, Tech. Rep. Version*, 1, 2004. 57

[11] Alexander Becher, Zinaida Benenson, and Maximillian Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. In *Proceedings of the Third International Conference on Security in Pervasive Computing*, SPC'06, pages 104–118, Berlin, Heidelberg, 2006. Springer-Verlag. 57

[12] Markus Becker, Koojana Kuladinithi, and Carmelita Gorg. Modelling and Simulating the Trickle Algorithm. In *Mobile Networks and Management*, volume 97 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 135–144. Springer Berlin Heidelberg, 2012. 153, 154, 155

[13] Shimshon Berkovits. How to broadcast a secret. In DonaldW. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 535–541. Springer Berlin Heidelberg, 1991. 109

[14] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, January 2005. 27

[15] Carsten Bormann, Mehmet Ersue, and Ari Keranen. Terminology for Constrained-Node Networks. RFC 7228, May 2014. 94

[16] Martina Brachmann, Sye Loong Keoh, Oscar Garcia-Morchon, and Sandeep S. Kumar. End-to-end transport security in the ip-based internet of things. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–5, July 2012. 62, 63

[17] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks. In *Proc. SenSys*, pages 307–320, New York, NY, USA, 2006. ACM. 19, 85, 87

[18] Angelo Capossele, Valerio Cervo, Gianluca De Cicco, and Chiara Petrioli. Security as a CoAP Resource: an Optimized DTLS Implementation for the IoT. In *Proc. ICC*, London, UK, 6-8 June 2015. IEEE. 62, 63

[19] CC2420. Texas Instruments. `www.ti.com/product/cc2420`. 10, 18, 41

[20] Yacine Challal, Hatem Bettahar, and Abdelmadjid Bouabdallah. Sakm: A scalable and adaptive key management approach for multicast communications. *SIGCOMM Comput. Commun. Rev.*, 34(2):55–70, April 2004. 129

[21] Yacine Challal, Hatem Bettahar, and Abdelmadjid Bouabdallah. A taxonomy of multicast data origin authentication: Issues and solutions. *Communications Surveys Tutorials, IEEE*, 6(3):34–57, Third 2004. 130

[22] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 197–213, May 2003. 56

[23] Tengfei Chang, Thomas Watteyne, Kristofer S.J. Pister, and Qin Wang. Adaptive synchronization in multi-hop TSCH networks. *Computer Networks*, 76:165 – 176, 2015. 24

[24] Feng Chen, Xiaolong Yin, Reinhard German, and Falko Dressler. Performance Impact of and Protocol Interdependencies of IEEE 802.15.4 Security Mechanisms. In *International Conference on Mobile Adhoc and Sensor Systems (MASS)*. IEEE, October 2009. 60, 61

[25] Simone Cirani, Gianluigi Ferrari, and Luca Veltri. Enforcing Security Mechanisms in the IP-Based Internet of Things: An Algorithmic Overview. *Algorithms*, 6(2):197–226, 2013. 64

[26] Titouan Coladon. Modeling the trickle algorithm to control overhead. Master's thesis, University Joseph Fourier, 2014. 156

[27] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013. 34, 36

[28] Roberta Daidone, Gianluca Dini, and Marco Tiloca. On Experimentally Evaluating the Impact of Security on IEEE 802.15.4 networks. In *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–6, June 2011. 60, 61, 71

[29] Giacomo de Meulenaer, François Gosset, François-Xavier Standaert, and Olivier Pereira. On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks. In *Networking and Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing,*, pages 580–585, Oct 2008. 61

[30] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008. 85, 86

[31] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, Nov 1976. 47

[32] Danny Dolev and Andrew C Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983. 56

[33] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical report, Swedish Institute of Computer Science (SICS), 2011. 19

[34] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455 – 462, nov. 2004. 30

[35] Adam Dunkels, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th Workshop on Embedded Networked Sensors*, EmNets '07, pages 28–32, New York, NY, USA, 2007. ACM. 72

[36] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 29–42, New York, NY, USA, 2006. ACM. 30

[37] Morris Dworkin. Recommendation for block cipher modes of operation: The ccm mode for authentication and confidentiality. *NIST Special Publication 800–38C*, 2004. 39, 110

[38] Amre El-Hoiydi. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 5, pages 3418–3423 vol.5, 2002. 19

[39] Barker Elaine. Recommendation for key management–part 1: General (revision 4). *Draft NIST Special Publication 800–57*, September 2015. 34, 48

[40] TR ETSI. 102 691 V1. 1.1 Machine-to-Machine communications (M2M). *Smart Metering Use Cases*, 2011. 99

[41] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000. 26, 97, 99

[42] HART Communication Foundation. WirelessHART Specification 75: TDMA Data-Link Layer, 2008. HCF_SPEC-75. 122

[43] Stefanie Gerdes, Olaf Bergmann, and Carsten Bormann. Delegated authenticated authorization for constrained environments. In *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*, pages 654–659, Oct 2014. 63

[44] Stefanie Gerdes, Olaf Bergmann, and Carsten Bormann. Delegated CoAP Authentication and Authorization Framework (DCAF). draft-gerdes-ace-dcaf-authorize-02. Work in progress, March 2015. 66

[45] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, New York, NY, USA, 2009. ACM. 153

[46] Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. A secure interconnection model for IPv6 enabled wireless sensor networks. *Wireless Days*, 2010. 61

[47] Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. On the Effectiveness of End-to-End Security for Internet-Integrated Sensing Applications. In *Proc. GreenCom*, pages 87–93. IEEE, 20-23 November 2012. 61, 62, 63, 85

[48] Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. Application-Layer Security for the WoT: Extending CoAP to Support End-to-End Message Security for Internet-Integrated Sensing Applications. *Wired/Wireless Internet Communication*, 7889:140–153, 2013. 64

[49] Jorge Granjal, Ricardo Silva, Edmundo Monteiro, Jorge Sa Silva, and Fernando Boavida. Why is IPSec a viable option for wireless sensor networks. In *MASS*, pages 802–807. IEEE, 2008. 61

[50] Vipul Gupta, Michael Wurm, Yu Zhu, Matthew Millard, Stephen Fung, Nils Gura, Hans Eberle, and Sheueling Chang Shantz. Sizzle: A standards-based end-to-end security architecture for the embedded internet. *Perv. Mob. Comp.*, 1(4):425–445, 2005. 62, 97

[51] Panu Hamalainen, Marko Hannikainen, and Timo D. Hamalainen. Efficient Hardware Implementation of Security Processing for IEEE 802.15.4 Wireless Networks. In *Midwest Symposium on Circuits and Systems*, volume 1, pages 484–487, August 2005. 46

[52] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006. 48, 49, 50

[53] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012. 66

[54] Klaus Hartke. Practical Issues with Datagram Transport Layer Security in Constrained Environments. draft-hartke-dice-practical-issues-01. Work in progress, April 2013. 111

[55] Klaus Hartke. Observing Resources in CoAP. draft-ietf-core-observe-16. Work in progress, December 2014. 29

[56] Sungmin Hong, Daeyoung Kim, Minkeun Ha, Sungho Bae, Sang Jun Park, Wooyoung Jung, and Jae-Eon Kim. SNAIL: an IP-based wireless sensor network approach to the internet of things. *Wireless Comm. IEEE*, 17(6):34–42, 2010. 62

[57] Lian Huai, Xuecheng Zou, Zhenglin Liu, and Yu Han. An Energy-Efficient AES-CCM Implementation for IEEE802.15.4 Wireless Sensor Networks. In *International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC)*, volume 2, pages 394–397, April 2009. 46

[58] Jonathan W. Hui and David E. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 81–94, New York, NY, USA, 2004. ACM. 153

[59] Jonathan W. Hui and Richard Kelsey. Multicast Protocol for Low power and Lossy Networks (MPL). draft-ietf-roll-trickle-mcast-12. Work in progress, June 2015. 153

[60] Rene Hummen, Hossein Shafagh, Shahid Raza, Thiemo Voigt, and Klaus Wehrle. Delegation-based Authentication and Authorization for the IP-based Internet of Things. In *Proc. of SECON*. IEEE, 2014. 63, 85

[61] Rene Hummen, Jan H. Ziegeldorf, Hossein Shafagh, Shahid Raza, and Klaus Wehrle. Towards Viable Certificate-based Authentication for the Internet of Things. In *Proc. HotWiSec Workshop*, pages 37–42, New York, NY, USA, 2013. ACM. 62, 63, 111, 113

[62] IEEE. 802.15.4-2011: IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), 2011. 18, 19, 41, 85, 143

[63] IEEE. 802.15.4e-2012: IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer, 2012. 10, 23, 57, 85

[64] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Emerging networking experiments and technologies*, pages 1–12. ACM, 2009. 98, 100

[65] Michael B. Jones. JSON Web Key (JWK). RFC 7517, May 2015. 65

[66] Michael B. Jones, John Bradley, and Nat Sakimura. JSON Web Signature (JWS). RFC 7515, May 2015. 65

[67] Michael B. Jones and Dick Hardt. The OAuth 2.0 Authorization Framework: Bearer Token Usage. RFC 6750, October 2012. 120

[68] Michael B. Jones and Joe Hildebrand. JSON Web Encryption (JWE). RFC 7516, May 2015. 65

[69] Taeho Jung, Xiang-Yang Li, Zhiguo Wan, and Meng Wan. Privacy preserving cloud data access with multi-authorities. In *INFOCOM, 2013 Proceedings IEEE*, pages 2625–2633, April 2013. 101

[70] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2):293–315, 2003. 56

[71] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1(2–3):293 – 315, 2003. Sensor Network Protocols and Applications. 58, 59

[72] Niels Karowski, Aline Carneiro Viana, and Adam Wolisz. Optimized Asynchronous Multichannel Discovery of IEEE 802.15.4-Based Wireless Personal Area Networks. *Transactions on Mobile Computing, IEEE*, 12(10):1972–1985, 2013. 144, 150

[73] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network security: private communication in a public world*. Prentice Hall Press, 2002. 34

[74] Sye Keoh, Sandeep Kumar, Oscar Garcia-Morchon, Esko Dijk, and Akbar Rahman. DTLS-based Multicast Security for Low-Power and Lossy Networks (LLNs). draft-keoh-dice-multicast-security-05. Work in progress, February 2014. 65

[75] Hamidreza Kermajani and Carles Gomez. Route change latency in low-power and lossy wireless networks using rpl and 6lowpan neighbor discovery. In *Computers and Communication (ISCC), 2011 IEEE Symposium on*, pages 937–942. IEEE, 2011. 151

[76] Hamidreza Kermajani and Carles Gomez. On the Network Convergence Process in RPL over IEEE 802.15. 4 Multihop Networks: Improvement and Trade-Offs. *Sensors*, 14(7):11993–12022, 2014. 153, 154, 155

[77] Hamidreza Kermajani, Carles Gomez, and Mostafa Hesami Arshad. Modeling the message count of the trickle algorithm in a steady-state, static wireless sensor network. *Communications Letters, IEEE*, 16(12):1960–1963, December 2012. 153, 154, 155, 156, 161

[78] Moazzam Khan, Fereshteh Amini, Jelena Mišić, and Vojislav B. Mišić. The cost of security: Performance of zigbee key exchange mechanism in an 802.15.4 beacon enabled cluster. In *Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference on*, pages 876–881, Oct 2006. 61

[79] Mikko Kohvakka, Jukka Suhonen, Mauri Kuorilehto, Ville Kaseva, Marko Hännikäinen, and Timo D. Hämäläinen. Energy-Efficient Neighbor Discovery Protocol for Mobile Wireless Sensor Networks. *Ad Hoc Networks, Elsevier*, 7(1):24 – 41, 2009. 147, 150

[80] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM Comput. Commun. Rev.*, 37(4):181–192, August 2007. 98

[81] Michael Koster, Ari Keranen, and Jaime Jimenez. Publish-Subscribe Broker for the Constrained Application Protocol (CoAP). draft-koster-core-coap-pubsub-02. Work in progress, July 2015. 119, 122

[82] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brunig, and Georg Carle. A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication. In *LCN*, pages 956–963. IEEE, 2012. 62, 63

[83] Sandeep S Kumar, Sye Loong Keoh, and Hannes Tschofenig. A Hitchhiker's Guide to the (Datagram) Transport Layer Security Protocol for Smart Objects and Constrained Node Networks. draft-ietf-lwig-tls-minimal-01. Work in progress., March 2014. 63

[84] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. Pearson Education, 4th edition, 2008. 16

[85] Chi-Anh La, Martin Heusse, and Andrzej Duda. Link reversal and reactive routing in Low Power and Lossy Networks. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, pages 3386–3390, Sept 2013. 30, 133

[86] RSA Laboratories. PKCS #1 v2.1. *RSA Cryptography Standard*, June 2002. 48

[87] Steven Lanzisera and Kristofer S.J. Pister. Theoretical and Practical Limits to Sensitivity in IEEE 802.15.4 Receivers. In *Electronics, Circuits and Systems, 2007. ICECS 2007. 14th IEEE International Conference on*, pages 1344–1347, Dec 2007. 9

[88] Yee Wei Law, Lodewijk van Hoesel, Jeroen Doumen, Pieter Hartel, and Paul Havinga. Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '05, pages 76–88, New York, NY, USA, 2005. ACM. 57

[89] Ilias Leontiadis, Christos Efstratiou, Cecilia Mascolo, and Jon Crowcroft. SenShare: transforming sensor networks into multi-application sensing infrastructures. In *Wireless Sensor Networks*, pages 65–81. Springer, 2012. 99

[90] Philip Levis, Thomas Heide Clausen, Jonathan Hui, Omprakash Gnawali, and JeongGil Ko. The Trickle Algorithm. RFC 6206, March 2011. 153, 154

[91] Philip Levis, Neil Patel, David E. Culler, and Scott Shenker. Trickle: a Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *NSDI*, volume 1, pages 15–28. USENIX Association, 2004. 136, 139, 154

[92] Philip Levis, Arsalan Tavakoli, and Stephen Dawson-Haggerty. Overview of Existing Routing Protocols for Low Power and Lossy Networks. draft-ietf-roll-protocols-survey-07. Work in progress, April 2009. 134

[93] Mingyan Li, Iordanis Koutsopoulos, and Radha Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1307–1315, May 2007. 57

[94] Linear Technology - Dust Networks. LTC5800-IPM. `http://www.linear.com/product/LTC5800-IPM`. 8, 10

[95] An Liu and Peng Ning. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 245–256, April 2008. 51, 110

[96] Fang Liu, Xiuzhen Cheng, and Dechang Chen. Insider attacker detection in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1937–1945, May 2007. 57

[97] Alessandro Ludovici, Anna Calveras, and Jordi Casademont. Forwarding techniques for IP fragmented packets in a real 6LoWPAN network. *Sensors*, Vol. 11:992–1008, 2011. 25

[98] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Gligor. MiniSec: a secure sensor network communication architecture. In *IPSN*, page 479. IEEE, 2007. 61

[99] Stefan Mahlknecht and Michael Bock. Csma-mps: a minimum preamble sampling mac protocol for low power wireless sensor networks. In *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*, pages 73–80, Sept 2004. 19

[100] David McGrew and Daniel V. Bailey. AES-CCM Cipher Suites for Transport Layer Security (TLS). RFC 6655, July 2012. 41

[101] Memsic. MicaZ mote. `http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf`. 10

[102] Alfred J. Menezes, Edlyn Teske, and Annegret Weng. Weak fields for ecc. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 366–386. Springer Berlin Heidelberg, 2004. 49

[103] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC press, 1996. 33, 34, 36, 37, 48

[104] Thomas M.M. Meyfroyt, Sem C. Borst, Onno J. Boxma, and Dee Denteneer. Data dissemination performance in large-scale sensor networks. *SIGMETRICS Perform. Eval. Rev.*, 42(1):395–406, June 2014. 153, 154, 155

[105] Jelena Mišić. Cost of secure sensing in IEEE 802.15.4 networks. *Wireless Communications, IEEE Transactions on*, 8(5):2494–2504, May 2009. 61

[106] Jelena Mišić, Shairmina Shafi, and Vojislav B. Mišić. Performance of a beacon enabled IEEE 802.15.4 cluster with downlink and uplink traffic. *Parallel and Distributed Systems, IEEE Transactions on*, 17(4), 2006. 150

[107] Nagendra Modadugu and Eric Rescorla. The design and implementation of datagram TLS. In *Proceedings of ISOC NDSS*, pages 1–13, 2004. 62, 63, 98

[108] Gabriel Montenegro, Nandakishore Kushalnagar, Jonathan Hui, and David Culler. Transmission of IPv6 packets over IEEE 802.15.4 networks. rfc 4944, 2007. 25

[109] Marcello Mura, Fabio Fabbri, and Maria Giovanna Sami. Modelling the power cost of security in Wireless Sensor Networks: The case of 802.15.4. In *Telecommunications, 2008. ICT 2008. International Conference on*, pages 1–8, June 2008. 61, 71

[110] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: Analysis & defenses. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, IPSN '04, pages 259–268, New York, NY, USA, 2004. ACM. 58

[111] Olaf Bergmann, TZI Uni Bremen. tinyDTLS, 2013. 111

[112] OpenMote-CC2538. `http://www.openmote.com/openmote-cc2538/`. 43

[113] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with COOJA. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641 –648, nov. 2006. 89, 110, 146

[114] Carlos Tadeo Ortega Otero, Jonathan Tse, and Rajit Manohar. AES Hardware-Software Co-design in WSN. In *International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 85–92. IEEE, May 2015. 46

[115] Maria Rita Palattella, Nicola Accettura, Xavier Vilajosana, Thomas Watteyne, Luigi Alfredo Grieco, Gennaro Boggia, and Mischa Dohler. Standardized Protocol Stack for the Internet of (Important) Things. *Communications Surveys Tutorials, IEEE*, 15(3):1389–1406, Third 2013. 1, 2, 9, 18, 23, 24, 26, 30

[116] Bogdan Pavković, Andrzej Duda, Won-Joo Hwang, and Fabrice Theoleyre. Efficient Topology Construction for RPL over IEEE 802.15.4 in Wireless Sensor Networks. *Ad Hoc Networks, Elsevier*, 2013. 139, 151

[117] Bogdan Pavković, Fabrice Theoleyre, and Andrzej Duda. Multipath Opportunistic RPL Routing over IEEE 802.15.4. In *Proceedings of the MSWiM Conference*, pages 179–186. ACM, 2011. 147, 151

[118] Adrian Perrig, Ran Canetti, J. Doug Tygar, and Dawn Song. The TESLA broadcast authentication protocol. *RSA CryptoBytes*, 5, 2002. 130

[119] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, June 2004. 58

[120] Kristofer S.J. Pister. Smart dust: Autonomous sensing and communication in a cubic millimeter. `http://robotics.eecs.berkeley.edu/~pister/SmartDust/`. 2

[121] Kristofer S.J. Pister. Single chip mote. Berkeley Sensor and Actuator Center, 2015. 11

[122] Kristofer S.J. Pister, Joseph M. Kahn, and Bernhard E. Boser. Smart dust: Wireless networks of millimeter-scale sensor nodes. *1999 UCB Electronics Research Laboratory Research Summary*, 1999. 1

[123] Akbar Rahman and Esko Dijk. Group Communication for the Constrained Application Protocol (CoAP). RFC 7390, October 2014. 29

[124] David R. Raymond and Scott F. Midkiff. Denial-of-service in wireless sensor networks: Attacks and defenses. *Pervasive Computing, IEEE*, 7(1):74–81, Jan 2008. 57

[125] Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, and Utz Roedig. Securing Communication in 6LoWPAN with Compressed IPsec. In *Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–8, June 2011. 61, 97

[126] Shahid Raza, Simon Duquennoy, Joel Höglund, Utz Roedig, and Thiemo Voigt. Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN. *Security and Communication Networks, Wiley*, 7(12):2654–2668, December 2014. 47, 60, 61

[127] Shahid Raza, Hossein Shafagh, Kasun Hewage, Rene Hummen, and Thiemo Voigt. Lithe: Lightweight Secure CoAP for the Internet of Things. *IEEE Sensors Journal*, 13(10):3711–3720, 2013. 62, 63, 85, 90, 97, 112

[128] Shahid Raza, Daniele Trabalza, and Thiemo Voigt. 6LoWPAN Compressed DTLS for CoAP. In *Distributed Computing in Sensor Systems (DCOSS),*

2012 IEEE 8th International Conference on, pages 287–289. IEEE, 2012. 62, 63, 97

[129] Eric Rescorla and Nagendra Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347, January 2012. 30, 62, 85, 86

[130] Mohsen Rezvani, Aleksandar Ignjatovic, Elisa Bertino, and Sanjay Jha. Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks. *Dependable and Secure Computing, IEEE Transactions on*, 12(1):98–110, Jan 2015. 56, 57

[131] Michael C. Richardson. 6tisch secure join using 6top. draft-richardson-6tisch–security-6top-04. Work in progress, November 2014. 58, 65

[132] Rodrigo Roman and Javier Lopez. Integrating wireless sensor networks and the internet: a security analysis. *Internet Research*, 19(2), 2009. 61

[133] Gabriele Romaniello. *Energy Efficient Protocols for Harvested Wireless Sensor Networks*. PhD thesis, Université de Grenoble, March 2015. xiii, 11, 12, 15

[134] Gabriele Romaniello, Emmanouil Potetsianakis, Olivier Alphand, Roberto Guizzetti, and Andrzej Duda. Fast and energy-efficient topology construction in multi-hop multi-channel 802.15.4 networks. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, pages 382–387, Oct 2013. 150

[135] Joseph Salowey, Hao Zhou, Pasi Eronen, and Hannes Tschofenig. Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 5077, 2008. 65

[136] Naveen Sastry and David Wagner. Security considerations for IEEE 802.15.4 networks. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 32–42. ACM, 2004. 60, 70

[137] Bruce Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. John Wiley & Sons, 2007. 34, 36, 48

[138] Bruce Schneier. `https://www.schneier.com/blog/archives/2015/06/the_secrecy_of_.html`, June 2015. 34

[139] Ludwig Seitz, Stefanie Gerdes, Goeran Selander, Mehdi Mani, and Sandeep S. Kumar. ACE use cases. draft-seitz-ace-usecases-05. Work in progress, September 2015. 65, 87, 107

[140] Ludwig Seitz, Göran Selander, and Christian Gehrmann. Authorization framework for the internet-of-things. In *WoWMoM*, pages 1–6. IEEE, 2013. 64

[141] Göran Selander, John Mattsson, Francesca Palombini, and Ludwig Seitz. Object Security for CoAP (OSCOAP). draft-selander-ace-object-security-02. Work in progress, June 2015. 66, 118, 129, 130

[142] Mohit Sethi, Jari Arkko, and Ari Keranen. End-to-end security for sleepy smart object networks. In *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*, pages 964–972, Oct 2012. 64

[143] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. 109

[144] Zach Shelby, Klaus Hartke, and Carsten Bormann. Constrained Application Protocol (CoAP). RFC 7252, June 2014. ix, 26, 27, 28, 29, 62, 63, 65, 85, 97, 107, 122, 129

[145] Robert W. Shirey. Internet Security Glossary, Version 2. RFC 4949, August 2007. 55

[146] Diana Smetters and Van Jacobson. Securing network content. *Technical Report TR-2009-1, Xerox Palo Alto Research Center-PARC*, 2009. 98

[147] Ohyoung Song and Jiho Kim. An efficient design of security accelerator for ieee 802.15.4 wireless senor networks. In *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pages 1–5, Jan 2010. 46

[148] STMicroelectronics. STM32L1 Series. `http://www.st.com/stm32l1`. 8, 12, 41

[149] Rene Struik. 6TiSCH Security Architectural Considerations. draft-struik-6tisch-security-considerations-01. Work in progress, January 2015. 58, 65

[150] Rene Struik, Yoshihiro Ohba, and Subir Das. 6TiSCH Security Architectural Elements, Desired Protocol Properties, and Framework. draft-struik-6tisch-security-architecture-elements. Work in progress, October 2014. 58, 65

[151] Swedish ICT. The Contiki OS. `http://www.contiki-os.org/`. 89

[152] Pawel Szalachowski and Tiffany Hyun-Jin Kim. Secure broadcast in distributed networks with strong adversaries. *Security and Communication Networks*, 2015. 109

[153] Pawel Szalachowski and Adrian Perrig. Lightweight protection of group content distribution. In *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security*, IoTPTS '15, pages 35–42, New York, NY, USA, 2015. ACM. 129

[154] International Telegraph and Telephone Consultative Committee. *CCITT Recommendation X. 800: Data Communication Networks: Open Systems Interconnection (OSI); Security, Structure and Applications: Security Architecture for Open Systems Interconnection for CCITT Applications*. International Telecommunication Union, 1991. 55

[155] TelosB motes. `http://www4.ncsu.edu/~kkolla/CSC714/datasheet.pdf`. 43

[156] Texas Instruments. CC2520. `www.ti.com/product/cc2520`. 10, 41

[157] Texas Instruments. CC2538. `http://www.ti.com/product/cc2538`. 8, 10, 41, 51

[158] Texas Instruments. MSP430F1x. `www.ti.com/MSP430`. 8

[159] Fabien Todeschini. *Dimensionnement energetique de reseaux de capteurs ultra-compacts autonomes en energie*. PhD thesis, Supelec, 2014. 13

[160] Fabien Todeschini, Christophe Planat, Patrizia Milazzo, Salvatore Tricomi, Pascal Urard, and Philippe Benabes. A nano quiescent current power management for autonomous wireless sensor network. In *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*, pages 815–818, Dec 2013. 13

[161] Joydeep Tripathi, Jaudelice C. de Oliveira, and Jean Philippe Vasseur. Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6687, October 2012. 141

[162] Hannes Tschofenig. The OAuth 2.0 Bearer Token Usage over the Constrained Application Protocol (CoAP). draft-tschofenig-ace-oauth-bt-01. Work in progress, March 2015. 66

[163] Hannes Tschofenig. The OAuth 2.0 Internet of Things (IoT) Client Credentials Grant. draft-tschofenig-ace-oauth-iot-01. Work in progress, March 2015. 66

[164] Hannes Tschofenig and Thomas Fossati. TLS/DTLS Profiles for the Internet of Things. draft-ietf-dice-profile-14. Work in progress, August 2015. 59, 65

[165] Hannes Tschofenig, Eve Maler, Erik Wahlstroem, and Samuel Erdtman. Authentication and Authorization for Constrained Environments Using OAuth and UMA. draft-maler-ace-oauth-uma-00. Work in progress, March 2015. 66

[166] Nicolas Tsiftes and Adam Dunkels. A database in every sensor. In *Conference on Embedded Networked Sensor Systems*, pages 316–332. ACM, 2011. 99

[167] Carlo Vallati and Enzo Mingozzi. Trickle-f: Fair broadcast suppression to improve energy-efficient route formation with the rpl routing protocol. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*, pages 1–9, Oct 2013. 153, 154, 155

[168] Luca Veltri, Simone Cirani, Stefano Busanelli, and Gianluigi Ferrari. A Novel Batch-based Group Key Management Protocol Applied to the Internet of Things. *Ad Hoc Networks*, 11:2724–2737, 2013. 129

[169] Aditya Vempaty, Onur Ozdemir, Kunal Agrawal, Hao Chen, and Pramod K. Varshney. Localization in wireless sensor networks: Byzantines and mitigation techniques. *Signal Processing, IEEE Transactions on*, 61(6):1495–1508, March 2013. 56

[170] John R. Vollbrecht, Pat R. Calhoun, Stephen Farrell, Leon Gommans, George M. Gross, Betty de Bruijn, Cees T.A.M. de Laat, Matt Holdrege, and David W. Spence. AAA Authorization Framework. RFC 2904, August 2000. 119

[171] Mališa Vučinić. Routing in IPv6 Sensor Networks. Master's thesis, École nationale supérieure d'informatique et de mathématiques appliquées, 2012. 134

[172] Mališa Vučinić, Bernard Tourancheau, and Andrzej Duda. Performance comparison of the RPL and LOADng routing protocols in a Home Automation scenario. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 1974–1979, April 2013. 151

[173] Arvinderpas S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 324–328, March 2005. 48

[174] Brett Warneke, Matt Last, Brian Liebowitz, and Kristofer S.J. Pister. Smart dust: Communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51, 2001. 8

[175] Thomas Watteyne, Steven Lanzisera, Ankur Mehta, and Kristofer S.J. Pister. Mitigating multipath fading through channel hopping in wireless sensor networks. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5, May 2010. 23

[176] Thomas Watteyne, Ankur Mehta, and Kristofer S.J. Pister. Reliability through frequency diversity: Why channel hopping makes sense. In *Proceedings of the 6th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, PE-WASUN '09, pages 116–123, New York, NY, USA, 2009. ACM. 23

[177] Thomas Watteyne, Xavier Vilajosana, Branko Kerkez, Fabien Chraim, Kevin Weekly, Qin Wang, Steven Glaser, and Kristofer S.J. Pister. Openwsn: a standards-based low-power wireless development environment. *Transactions*

on Emerging Telecommunications Technologies, 23(5):480–493, 2012. 30, 40, 79

[178] Tim Winter, Pascal Thubert, Anders Brandt, Jonathan W. Hui, Richard Kelsey, Philip Levis, Kristofer S.J. Pister, Rene Struik, Jean Philippe Vasseur, and Roger K. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012. 139, 141, 142, 143, 153

[179] WisMote. `http://wismote.org`. 50, 110

[180] Alper Yegin and Zach Shelby. CoAP Security Options. draft-yegin-coap-security-options-00. Work in progress, October 2011. 65

[181] Zhou Yiming, Yang Xianglong, Guo Xishan, Zhou Mingang, and Wang Liren. A Design of Greenhouse Monitoring Control System Based on ZigBee Wireless Sensor Network. In *WiCom*, pages 2563–2567, 2007. 147

[182] Christopher D. Young. Intel developer forum. `http://intelstudios.edgesuite.net/idf/2015/sf/megasession/150818_cy/index.html`, August 2015. 1

[183] Yang Yu, Loren J. Rittle, Vartika Bhandari, and Jason B. LeBrun. Supporting concurrent applications in wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 139–152, New York, NY, USA, 2006. ACM. 153

[184] ZigBee Alliance. ZigBee Smart Energy Profile 2. 73, 76

[185] ZigBee Alliance. ZigBee Specification. 2, 22