



Politecnico
di Bari

Repository Istituzionale dei Prodotti della Ricerca del Politecnico di Bari

From Security to Localization: Advanced Services for Large-Scale IoT Networks

This is a PhD Thesis

Original Citation:

From Security to Localization: Advanced Services for Large-Scale IoT Networks / Sciancalepore, Savio. - (2017).
[10.60576/poliba/iris/sciancalepore-savio_phd2017]

Availability:

This version is available at <http://hdl.handle.net/11589/98114> since: 2017-03-24

Published version

DOI:10.60576/poliba/iris/sciancalepore-savio_phd2017

Publisher: Politecnico di Bari

Terms of use:

(Article begins on next page)



DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING
ELECTRICAL AND INFORMATION ENGINEERING

PHD PROGRAM
S.S.D. ING-INF/03

Final Dissertation

From security to localization: advanced services for large-scale IoT networks

by

Savio SCIANCALEPORE

Referees:

Prof. Periklis Chatzimisios

Prof. Veljko Pejovic

Supervisor:

Prof. Ing. Gennaro Boggia

Coordinator of the Ph.D Program:

Prof. Ing. Vittorio Passaro

CYCLE XXIX, 2014-2016



DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING
ELECTRICAL AND INFORMATION ENGINEERING
PHD PROGRAM
S.S.D. ING-INF/03

Final Dissertation

From security to localization: advanced services for large-scale IoT networks

by

Savio SCIANCALEPORE

Referees:

Prof. Periklis Chatzimisios

Prof. Veljko Pejovic

Supervisor:

Prof. Ing. Gennaro Boggia

Coordinator of the Ph.D Program:

Prof. Ing. Vittorio Passaro

CYCLE XXIX, 2014-2016

To my family, that supported me since from the beginning of my studies in dedicating time and efforts in studying and increasing my knowledge.

To my dear girlfriend, Marianna, who supported me in this path, who motivated me during dark periods and who strongly encouraged me in becoming a better person and a better researcher.

Acknowledgements

I would like to acknowledge my tutor, Prof. Ing. Gennaro Boggia, and my colleague, Giuseppe Piro, for all the efforts spent in assisting me on developing each of the research topic included in this thesis.

Contents

List of Figures	ix
List of Tables	xv
Introduction: Dissertation Overview	xvii
Personal Scientific Contributions	xxi
1 The Internet of Things technology	1
1.1 The rise of the Internet of Things	1
1.2 Applications of IoT technology	3
1.3 Toward the definition of a Standard Protocol Stacks for Internet of Things (IoT) communications	6
1.3.1 ISA 100.11a and WirelessHART specifications	7
1.3.2 Zigbee protocol stack	9
1.3.2.1 IEEE 802.15.4-2003	10
1.3.3 ZigBee IP	12
1.3.4 Bluetooth technology	13
1.3.5 6tisch protocol stack	15
1.3.5.1 Physical layer: IEEE 802.15.4-2006 standard	16
1.3.5.2 MAC layer: IEEE 802.15.4e-2012 amendment	17
1.3.5.3 6top and 6tisch-related activities	19
1.3.5.4 6LoWPAN Adaptation layer	21
1.3.5.5 Network layer: RPL protocol	21
1.3.5.6 Transport layer: UDP Protocol in IoT technology . . .	22
1.3.5.7 Application layer: the CoAP protocol	23

CONTENTS

1.4	Real IoT products	24
1.4.1	Hardware Devices	24
1.4.1.1	TelosB	24
1.4.1.2	Zolertia Z1	25
1.4.1.3	OpenMote-CC2538	25
1.4.2	Protocol stacks for the IoT	26
1.4.2.1	OpenWSN	26
1.4.2.2	Contiki	31
1.4.2.3	RIOT	32
2	Securing IoT systems	33
2.1	IoT security requirements	33
2.2	Practical limitations in real IoT systems	34
2.3	Challenges for a secure IoT	35
2.4	Security Algorithms for IoT	38
2.4.1	Symmetric Cryptography	38
2.4.1.1	The Advanced Encryption Standard (AES) algorithm	39
2.4.2	Asymmetric Cryptography	40
2.4.2.1	RSA	42
2.4.2.2	Diffie-Hellman Key Exchange	43
2.4.3	Elliptic Curve Cryptography	44
2.4.3.1	Elliptic Curve Diffie Hellman	45
2.5	The evolution of security systems for IoT	46
2.5.1	Security in WirelessHART and ISA 100.11a	46
2.5.2	Bluetooth security	48
2.5.3	Security in Zigbee IP	49
2.6	State of the art on security protocols for IoT networks	51
2.6.1	MAC layer security	51
2.6.1.1	Security in IEEE 802.15.4	51
2.6.2	Layer-2 security: literature review	58
2.6.2.1	Link-layer security overhead	58
2.6.2.2	Standardization Activities	60
2.6.2.3	Key Management Protocols	61

2.6.3	Application-layer security in IoT networks	61
3	Advanced MAC-layer security functionalities for large-scale IoT networks	65
3.1	Implementation of IEEE 802.15.4 security procedures in OpenWSN protocol stack	66
3.1.1	Implementation details	67
3.1.2	Usage of the developed module	70
3.2	Link-layer security in TSCH networks	71
3.2.1	Time-slot duration in the TSCH mode	72
3.2.2	The hardware support for security in TelosB and OpenMote-CC2538	74
3.2.3	Software Implementation	75
3.2.4	Hardware Implementation	76
3.2.5	Hybrid Implementation	76
3.2.6	Performance Evaluation	76
3.2.6.1	Goals and Methodology	77
3.2.6.2	Memory Footprint	79
3.2.6.3	Minimal Slot Length	79
3.2.6.4	Energy consumption	84
3.2.6.5	Final considerations on high-level application design	85
3.3	LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT	86
3.3.1	Details on the proposed security framework	88
3.3.1.1	Security configurations	89
3.3.1.2	Security Configuration Manager	92
3.3.1.3	Bootstrap Manager	93
3.3.1.4	Key Negotiation Manager	94
3.3.1.5	Details on the Global Security Levels Table	98
3.3.2	Performance evaluation and security analysis	98
3.3.2.1	Security Analysis	99
3.3.2.2	Preliminary investigation	103
3.3.2.3	Experimental settings	105

CONTENTS

3.3.2.4	Time and energy needed for configuring security services	107
3.4	A real-world application: TLSensing	109
3.4.1	TLSensing in a nutshell	113
3.4.1.1	Hardware details	116
3.4.1.2	Firmware and software details	116
3.4.2	TLSensing at work	117
3.5	EXCHANge: a solution to Key-Establishment for IoT leveraging Channel Anonymity	119
3.5.1	Scenario and adversary model	120
3.5.2	Key sharing based on channel anonymity	121
3.5.3	The EXCHANge key-establishment protocol	123
3.5.3.1	Definitions and assumptions	123
3.5.3.2	Finite state machine model	124
3.5.3.3	The EXCHANge algorithm	126
3.5.3.4	Implementation of the FSM	126
3.5.3.5	Collisions and Lost packets	128
3.5.4	Performance	130
3.5.5	Security Analysis: guessing the secret key	133
3.5.5.1	Mitigating the RSSI-based source identification	134
3.5.6	Energy analysis	139
3.5.6.1	Discussion and Remarks	141
4	Security solutions at the Application Layer for the IoT	143
4.1	Key Management Protocol with Implicit Certificates for IoT systems	144
4.1.1	Background: ECQV implicit certificates	145
4.1.2	Key Management Protocol	146
4.1.3	Security considerations	148
4.1.4	Implementation	149
4.1.5	Experimental performance assessment	151
4.2	Authorization in IoT systems	153
4.2.1	The OAuth 2.0 authorization framework	154
4.2.2	IETF token formats	156
4.2.3	The proposed OAuth-IoT framework	156

4.2.3.1	Resource management at the Resource Server	158
4.2.3.2	Sequence diagram for the resource access procedure . .	159
4.2.3.3	Details on tokens	160
4.2.3.4	Operations performed by the Resource Server	161
4.2.4	OAuth-IoT Testbed at work	163
4.3	The symbIoTe H2020 project	165
4.3.1	symbIoTe high-level description	166
4.3.2	Attribute-Based Access Control in federated IoT platforms . . .	168
4.3.2.1	Architecture	168
4.3.2.2	Main security rationale	169
4.3.2.3	Component description	171
4.3.2.4	Sequence Diagrams	172
4.3.2.5	Planned implementation	175
5	Time-based wireless localization techniques	181
5.1	Time-based localization logic	181
5.2	System Overview	181
5.3	Previous and related works	184
5.4	The system model	187
5.4.1	Notation and assumptions	187
5.5	Estimation of Clock Error Terms - Synchronization	190
5.6	Generalized Localization Algorithms	192
5.6.1	Preliminaries	192
5.6.2	TOA formulation	193
5.6.3	TDOA formulation	193
5.6.4	S-TOA formulation	195
5.6.5	Equality of TDOA and S-TOA	195
5.6.6	On the equivalence between the TOA and TDOA/S-TOA	196
5.6.7	A first look at resolution cost	197
5.6.8	Particularization to a static emitter (localization)	199
5.7	Simulation analysis	200
5.7.1	Scenario	200
5.7.2	Comparison with range-based DTDOA for static emitter	201

CONTENTS

5.7.3	Results with moving node	201
5.8	Experimental Results	203
5.8.1	Testbed setup	203
5.8.2	Experiment #1: comparison with range-based DTDOA localization for a static node	208
5.8.3	Experiment #2: moving node	208
5.8.4	Experiment #3: moving node with refined timestamps	209
5.9	Migration to IoT devices	211
6	Conclusions and Future Research Directions	213
	References	217

List of Figures

1.1	Evolution of number of IoT devices and world population in the last decade.	2
1.2	High-level view of IoT application domains.	3
1.3	High-level view of the Zigbee Protocol Stack.	9
1.4	Flow diagram of the CSMA-CA protocol.	11
1.5	The SuperFrame Structure.	12
1.6	ZigBee-IP protocol stack.	13
1.7	Big picture of the Bluetooth protocol stack.	14
1.8	Communication stack integrated in the <i>TL Sensing</i> platform.	15
1.9	TelosB board.	24
1.10	Z1 board.	25
1.11	OpenMote-CC2538 board.	25
1.12	The OpenWSN default protocol stack.	28
2.1	Establishing a secured communication in ZigBee IP specifications. . . .	50
2.2	The Auxiliary Security Header Structure.	53
2.3	CCM* allows every frame to be encrypted and/or authenticated. . . .	57
3.1	The OpenWSN default protocol stack.	67
3.2	The OpenWSN modified portion of the stack.	68
3.3	The minimal duration of a timeslot in an IEEE802.15.4e TSCH network depends on how long link-layer security operations take.	73
3.4	ROM and RAM Footprint of the implementations for TelosB mote. . . .	80
3.5	ROM and RAM Footprint of the implementations for OpenMote-CC2538 mote.	80

LIST OF FIGURES

3.6	Minimum timeslot duration for the TelosB mote.	82
3.7	Minimum timeslot duration for the OpenMote-CC2538 mote.	83
3.8	Energy consumed in a single timeslot for the TelosB mote.	84
3.9	Energy consumed in a single timeslot for the OpenMote-CC2538 mote.	85
3.10	General scheme of Lightweight sScheme for IoT secUre communicationS (LICITUS).	89
3.11	Message sequence chart of the designed Key Management Protocol (KMP) procedure.	95
3.12	Screen-shots of the output provided by the ProVerif automatic tool a) with the anonymous DH and the Master Key secret b) with the anonymous DH and the Master Key widely known and c) with the certified DH and the Master Key widely known.	103
3.13	Energy measurement system.	107
3.14	Time required for configuring security services in: (a) star, (b) chain, and (c) binary tree topologies.	110
3.15	Energy consumptions for configuring security services in: (a) star, (b) chain, and (c) binary tree topologies.	111
3.16	Network architecture.	114
3.17	TLSensing web page.	115
3.18	Interaction between the user and the measurement system.	117
3.19	Temperature and relative humidity acquired in an indoor environment during a single day.	118
3.20	Temperature and relative humidity acquired in an outdoor environment during a single day.	118
3.21	EXCHANge scenario.	121
3.22	Toy example of the EXCHANge protocol: $\mathcal{A}$ and $\mathcal{B}$, randomly choose a secret key, e.g., $K_A = [0, 0]$ and $K_B = [1, 1]$, and then they randomly interleave their transmissions: $[\mathcal{A}, \mathcal{B}, \mathcal{A}, \mathcal{B}]$. The transmitted bit is complemented when the transmitter is $\mathcal{A}$, while it is transmitted as it is when the transmitting source is $\mathcal{B}$	122
3.23	Channel anonymity: as soon as the anonymity of the transmitter is protected, $\mathcal{E}$ cannot deterministically decide whether to complement the received bit and therefore the secrecy of the transmitted bit is protected.	123

3.24	EXCHANge protocol: packet format for the IEEE 802.15.4e standard. The source and destination addresses have been sanitized, i.e., set all to zeros, and the least significant bit (LSB) of the MAC body is set accordingly to Eq. (3.16).	124
3.25	EXCHANge timeline: both $\mathcal{A}$ and $\mathcal{B}$ start in <i>sleep</i> mode, and switch to <i>RX</i> mode after 12 tics. Subsequently, both $\mathcal{A}$ and $\mathcal{B}$ schedule a random wait period: the first one ($\mathcal{A}$) reaching the period's end, it switches to the <i>TX</i> status initiating the transmission. Conversely, the second one ($\mathcal{B}$) remaining in the <i>RX</i> status, it receives the message. Eventually, the receiver ($\mathcal{B}$) acknowledges the packet reception with an <i>ack</i> message. . .	128
3.26	Time to run the EXCHANge protocol and generating a shared bit-string of 128 bits as a function of the distance between the devices. The error-bars show the quantile 5, 50, and 95 related to 20 runs of the protocol. .	130
3.27	Time to generate a bit-string of 128 bits varying the number of slots dedicated to the EXCHANge protocol in the slot-frame. A slot-frame size spanning in the interval $[11, \dots, 101]$ is assumed.	132
3.28	Time to transmit a bit-string of different sizes varying the number of dedicated slots (between 1 up to 10) in a slot-frame of 11 slots (default value for the OpenWSN protocol stack).	132
3.29	RSSI-based source identification: $\mathcal{E}$ measures the received signal strengths $R_{\mathcal{A}}$ and $R_{\mathcal{B}}$ from $\mathcal{A}$ and $\mathcal{B}$, respectively, and infers on the current transmitting source by looking at the different levels of the received power: (a) $R_{\mathcal{A}} \geq R_{\mathcal{B}}$, $R_{\mathcal{A}} \approx R_{\mathcal{B}}$, and $R_{\mathcal{A}} \leq R_{\mathcal{B}}$	134
3.30	Probability mass function associated to the received signal strengths experienced by $\mathcal{E}$ during real measurements featuring the scenarios of Figure 3.29.	135
3.31	Experimental set-up: $\mathcal{A}$ and $\mathcal{B}$ are placed at 6 meters far away while $\mathcal{E}$ is moved from 3m up to 0.5m close to $\mathcal{A}$	135
3.32	RSSI-based source identification: $\mathcal{E}$ splits the set of experienced RSS values into two subsets. Then, she decides that the smaller values (on the left of the red bar) are from the farther device ($\mathcal{B}$) while the bigger ones (on the right of the red bar) are from the closer node ($\mathcal{A}$).	137

LIST OF FIGURES

3.33	$\mathcal{E}$ guessing capabilities: quantile 5, 50, and 95 associated to the number of bits not-guessed by $\mathcal{E}$ (secret bits) during the RSSI-based source identification attack on a bit-string of 128 bit varying the distance between $\mathcal{E}$ and $\mathcal{A}$ as depicted in Figure 3.31.	138
3.34	Number of maximum transmissions requested to establish at least 80 secret bits as a function of the distance between $\mathcal{A}$ and $\mathcal{E}$	138
3.35	Set-up to measure the energy consumption.	139
3.36	Current drain of one run of the EXCHANge protocol: the device initially behaves as a transmitter and subsequently as a receiver.	140
3.37	Probability mass function associated to the current drain sampled during multiple runs of the EXCHANge protocol.	140
4.1	X.509-compliant implicit certificate format, main fields, from [1].	146
4.2	Key negotiation protocol.	147
4.3	Computational load of atomic operations with explicit X.509 certificates, PEM format.	152
4.4	Computational load of atomic operations with explicit X.509 certificates, DER format.	152
4.5	Computational load of atomic operations with implicit X.509 certificates.	153
4.6	Airtime consumption.	153
4.7	Protocol flow for the OAuth 2.0 access control framework.	155
4.8	Reference architecture.	157
4.9	Message Sequence Chart for a client accessing to a resource on the Resource Server (RS).	159
4.10	Flow diagram of the operations executed by the RS when a new HTTP request arrives from the Internet.	162
4.11	GUI of the OAuth-IoT testbed.	164
4.12	Latencies measurements.	165
4.13	Sketch of the <i>symbIoTe</i> architecture [2].	167
4.14	System Architecture.	169
4.15	An application registered in a given IoT platform wants to access resources produced within its home IoT platform.	173

4.16	An application registered in the mediator space wants to access resources in an IoT platform.	174
4.17	Multi-Domain Access Rights Composition.	176
4.18	Details of the content of (a) Macaroons and (b) JWT tokens for the designed architecture.	179
5.1	Reference Scenario.	182
5.2	Graphical representation of dataset pre-processing. Stage (c) is needed only for TDOA.	185
5.3	Workflow of the proposed method.	190
5.4	Number of iterations of dynamic models of TOA, TDOA, and S_TOA algorithms in the asynchronous case with 20% packet losses.	198
5.5	Workflow of the range-based DTDOA method in [3].	200
5.6	Reference topology with $N = 5$ anchors.	201
5.7	Final localization error for static blind node (asynchronous scenario with $\ell = 0$).	202
5.8	ECDF of horizontal error on mid-point for different loss intensities ($\ell \in \{0, 0.2, 0.4\}$).	204
5.9	Module of the estimated speed versus module of the true speed in the asynchronous case with 20% packet losses.	205
5.10	Angle between the vector of estimated speed and the vector of true speed versus module of the true speed in the asynchronous case with 20% packet losses.	205
5.11	ECDF of time-of-arrival differences between two receiving anchors after compensating for clock drift and offset. Top plot: raw timestamps before CSI correction. Bottom plot: after CSI correction.	207
5.12	Two of the 6-anchor topologies used for comparison with range-based DTDOA in WiFi testbed experiments.	207
5.13	Experimental performances of different localization schemes for static node: ECDF of absolute error over 54 trials.	208
5.14	Actual trajectory (blue) vs. Estimated trajectory (red) with TOA in three sample experiments.	209

LIST OF FIGURES

5.15 Actual trajectory (blue) vs. Estimated trajectory (red) with TOA in three sample experiments with CSI correction.	210
---	-----

List of Tables

1	Overview of the scientific publications included in this PhD dissertation	xxiv
1.1	Sensors already available on the OpenBattery.	27
1.2	Platform running the OpenWSN protocol stack.	30
2.1	Threats in the life-cycle of an IoT device.	35
2.2	Different security levels provided by the IEEE802.15.4 standard.	52
2.3	Security-related attributes defined in IEEE 802.15.4.	54
3.1	Description of IEEE 802.15.4 MAC security attributes implemented in OpenWSN.	70
3.2	Implementation Strategies.	74
3.3	The different steps in the tuning procedure.	78
3.4	Relationship between high-level security configurations and IEEE 802.15.4 security options.	91
3.5	Practical configurations of the Globals Security Levels Table.	99
3.6	Memory requirements.	105
3.7	EXCHANge: Statuses of the finite state machine.	129
3.8	Protocol consumption assuming a secret key of 80 bits and estimation of the battery life percentage consumed by one run of the protocol assuming the mote is powered by two AA batteries.	141
4.1	Registered Claim Names of the JWT and related description.	178

LIST OF TABLES

Introduction: Dissertation Overview

The last decade has been the witness of the explosion of the Internet of Things (IoT) phenomenon. Billions of smart and tiny devices are hitting the market, equipped with sensing and actuating capabilities and the possibility to interact with them over the big Internet. This paves the way to a variety of real innovative applications in different contexts, such as environmental monitoring, smart building, home automation, health care, logistics and energy management, to name a few. Moreover, the innovations and step-ups they provide, especially in industrial applications, is creating and enabling a wide set of advanced services, indicated as the forth industrial revolution, Industry 4.0, that will leverage on the new specific paradigm of the Industrial Internet of Things (IIoT). Also, major leading companies of the sector expect an enormous growth of the technology, up to the ambitious threshold of 50 billion of smart connected devices in 2020.

In this ebullient context, the need for developing accurate and tailored services, resized on the requirements and features of the constrained devices at the basis of IoT technology, is still an incumbent hurdle.

From one side, security aspects stand in a prominent position. The unique features of the involved devices require, in fact, the design of tailored routines and protocols, able to minimally affect the regular communications, operations and the execution of the standard protocol stack.

From the other side, the deployment of mobile sensor nodes requires the presence of smart and lightweight localization and tracing techniques, able to provide step-by-step the position of the sensing entities without affecting the execution of their fundamental tasks and the lifetime of the battery-powered devices.

0. INTRODUCTION: DISSERTATION OVERVIEW

Starting from these premises, the present work wraps around the main topic of providing advanced services for IoT networks, with the aim of affecting the normal network operations only in a minimal part.

With the focus on security aspects, the procedures and services recommended by one of the most important standard for IoT protocol stack, IEEE 802.15.4e, have been implemented. The overhead of link-layer security in IIoT networks has been evaluated, in terms of minimal timeslot length and memory footprint. The provided implementation of link-layer security covers a range of hardware platforms, exploring different hardware/software implementation strategies. Through an extensive measurement campaign, the advantage of hardware acceleration for link-layer security have been precisely quantified and, furthermore, the impact that the resulting timeslot duration has on both high-level application design and energy consumption have been investigated.

Moreover, starting from a careful inquiry about the gaps left open by the MAC layer standard, a lightweight and standard compatible framework, namely *LICITUS*, has been developed, with the aim of providing a wide variety of security configurations in homogeneous and heterogeneous scenarios, integration support for adaptation to dynamic networks, lean and scalable initialization functionalities, a lightweight Key Management Protocol (KMP) and resilience to several attacks.

Furthermore, starting from the KMP developed in the *LICITUS* framework, an advanced version has been designed, integrating implicit Elliptic Curve Qu-Vanstone (ECQV) certificates with a standard Elliptic Curve Diffie-Hellman (ECDH) exchange. From a security perspective, it performs authentication and key derivation between any two peers in IIoT network. As confirmed by a proof-of-concept implementation and relevant experimental results, this proposed KMP guarantees the minimal airtime consumption with respect to conventional approaches, while providing also robust key negotiation, fast re-keying, and efficient protection against replay attacks.

For scenarios characterized by particular constraints, Physical-layer security techniques have been evaluated and integrated within the IEEE 802.15.4e standard. The resulting protocol, namely EXCHANGe, does not leverage any existing secret in the devices' memory nor uses any public key cryptography scheme to reach an agreement between involved devices. The EXCHANGe protocol has been also implemented in the OpenWSN protocol stack and tested on the OpenMote-CC2538 architecture, repre-

senting the first real-world implementation of a key-establishment protocol based on channel anonymity.

The research activity about security has also covered the authorization problem, which is quickly emerging as one of the most important barrier to protect against external threats coming from the public Internet. Within the context of the *symbIoTe* H2020 European project, an authorization architecture based on the widely known OAuth 2.0 authorization framework has been proposed. Thanks to security services provided by the OAuth 2.0 Authorization Server deployed at the edge of the IoT network, the proposed solution assures the precise identification of all accesses to resources exposed by the smart devices, while not affecting in any way their computational burden.

As regarding localization solutions, a 4-months period of research abroad at the Faculty of Computer and Information Science in Ljubljana (Slovenia) have been conducted in order to study and provide a lightweight off-line algorithm that addresses the problem of estimating the position and velocity of a radio transmitter moving with a constant, unknown speed. The provided technique is based on arrival timestamps gathered by fixed sensor nodes and it is completely asynchronous in its logic, because no assumption is made about any time synchronization between the involved devices. The performances of this technique have been investigated not only through a wide set of Monte-Carlo simulations, but also using real data provided by off-the-shelf devices, demonstrating its feasibility for adoption in specific IoT services.

Moreover, other research activities, strictly connected to those aforementioned, have been carried out during the PhD work.

A complete list of produced scientific contributions will be presented immediately after this introduction.

To conclude, a brief description of the structure of this thesis is provided below:

- Chapter 1: *The Internet of Things technology*. It discusses the main features of the upcoming IoT technology, with a particular focus on the standardized protocol stack and the variety of hardware devices specifically designed for such applications.
- Chapter 2: *Security issues in IoT networks*. It provides an extensive summary of the state of the art in the IoT security area, discussing also the most adopted

0. INTRODUCTION: DISSERTATION OVERVIEW

solutions and algorithms from a theoretic perspective.

- Chapter 3: *MAC-layer security in IoT networks*. In this chapter the implementation of security procedures, services and algorithm recommended by the IEEE 802.15.4 standard is detailed, as well as a lightweight security framework that integrates and smartly uses such services.
- Chapter 4: *Security solutions at the application layer for the IoT*. It discusses all the solutions specifically designed for addressing security issues at the application layer, starting from a lightweight KMP guaranteeing the minimal airtime consumption, up to advanced solutions addressing the authorization problem.
- Chapter 5: *Time-based wireless localization techniques*. It focuses on an algorithm developed to estimate and track, time-by-time, the position of a node moving at a constant, unknown velocity. The performance evaluation of the algorithm is described, both in a simulated environment and through real data, along with the advantages derived by its adoption in the IoT context.

Personal Scientific Contributions

Scientific contributions leading to publications during the PhD work are listed in what follows. They have been accepted for publication in international journals and conferences or they have been submitted and are still waiting for acceptance.

An overview of their focus and their location within this PhD dissertation is included in Tab. 1.

International Journals

1. Savio Sciancalepore, Giuseppe Piro, Gennaro Boggia and Giuseppe Bianchi, Public Key Authentication and Key Agreement in IoT devices with minimal airtime consumption, accepted for publication in IEEE Embedded System Letters (ESL), doi: 10.1109/LES.2016.2630729.
2. Savio Sciancalepore, Giuseppe Piro, Elvis Vogli, Gennaro Boggia, Luigi Alfredo Grieco and Giuseppe Cavone, LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT, Elsevier Computer Networks (COMNET), vol. 108, no. 24, pp. 66 - 77, Aug., 2016, doi: 10.1016/j.comnet.2016.08.003.
3. Savio Sciancalepore, Mališa Vučinić, Giuseppe Piro, Gennaro Boggia and Thomas Watteyne, Link-layer security in TSCH networks: effect on slot duration, Wiley Transactions on Emerging Telecommunications Technologies (ETT), Jul., 2016, doi: 10.1002/ett.3089.
4. Fabio Ricciato, Savio Sciancalepore and Gennaro Boggia, Tracing a linearly moving node from asynchronous time-of-arrival measurements, IEEE Communications Letters, vol. 20, no. 9, pp., Jul. 2016, doi: 10.1109/LCOMM.2016.2584600.

0. PERSONAL SCIENTIFIC CONTRIBUTIONS

5. Savio Sciancalepore, Giuseppe Piro, Gennaro Boggia and Luigi Alfredo Grieco, Application of IEEE 802.15.4 security procedures in OpenWSN protocol stack, IEEE Standards Education eMagazine (eZine), vol. 4 , no. 2, pp., Third Quarter. 2014.

International Conferences

1. Savio Sciancalepore, Michal Pilc, Svenja Schröder, Giuseppe Bianchi, Gennaro Boggia, Marek Pawlowski, Giuseppe Piro, Marcin Plòciennik and Hannes Weisgrab, Attribute-Based Access Control scheme in federated IoT platforms, accepted for publication in the post-proceedings at the 2nd Workshop on Interoperability and Open-Source solutions for the IoT (InterOSS-IoT), Springer Lecture Notes on Computer Science (LNCS series), Stuttgart, Nov.6, 2016.
2. Savio Sciancalepore, Angelo Caposelle, Giuseppe Piro, Gennaro Boggia, and Giuseppe Bianchi, Key Management Protocol with Implicit Certificates for IoT systems, in Proceedings of the ACM Workshop on IoT challenges in Mobile and Industrial Systems (IoT-Sys), Florence (Italy), May 21, 2015, pp. 37 - 42. doi: 10.1145/2753476.2753477.
3. Savio Sciancalepore, Giuseppe Piro, Francesco Bruni, Enrico Nasca, Gennaro Boggia and Luigi Alfredo Grieco, An IoT-based measurement system for aerial vehicles, in Proc. of IEEE Workshop on Metrology for Aerospace (MetroAeroSpace), Benevento, Jun. 4-5, 2015, pp. 245 - 250. doi: 10.1109/MetroAeroSpace.2015.7180662.
4. Savio Sciancalepore, Giuseppe Piro, Elvis Vogli, Gennaro Boggia and Luigi Alfredo Grieco, On securing IEEE 802.15.4 networks through a standard compliant framework, in Proc. of IEEE Euro Med Telco Conference (EMTC), 2014, Naples, Nov. 12-15, 2014, pp. 1 - 6. doi: 10.1109/EMTC.2014.6996650.

National Conferences

1. Savio Sciancalepore, Giuseppe Piro, Elvis Vogli, Gennaro Boggia and Luigi Alfredo Grieco, On securing IEEE 802.15.4 networks through a standard compliant framework, presented in the Annual Workshop of GTTI (Gruppo nazionale Telecomunicazioni e Tecnologie dell'Informazione), Palermo (Italy), 18-20 Jun., 2014.

-
2. Savio Sciancalepore, Angelo Capossele, Giuseppe Piro, Gennaro Boggia, and Giuseppe Bianchi, On the Design of Lightweight Link-layer Security Mechanisms in IoT Systems, presented at the 12th Italian Networking Workshop, Cavalese, Italy - January 14 - 16, 2015
 3. Savio Sciancalepore, Daniele Caldarola, Giuseppe Piro Gennaro Boggia, Giuseppe Bianchi, On the road towards authenticated and authorized access in federated IoT platforms, presented at the 14th Italian Networking Workshop, Falcade, Italy - January 11 - 13, 2017.
 4. Savio Sciancalepore, Fabio Ricciato, Francesco Gringoli, Niccolò Facchi, Gennaro Boggia, Generalised localisation of a non-cooperative source from asynchronous TOA and TDOA measurements, presented at the 14th Italian Networking Workshop, Falcade, Italy - January 11 - 13, 2017.

Submitted Papers waiting for revision

1. Fabio Ricciato, Savio Sciancalepore, Francesco Gringoli, Niccolò Facchi, Gennaro Boggia, Generalised localisation of a non-cooperative source from asynchronous TOA and TDOA measurements, submitted to IEEE Transactions on Mobile Computing (TMC).
2. Savio Sciancalepore, Gabriele Oligeri, Giuseppe Piro, Roberto Di Pietro and Gennaro Boggia, EXchange: a solution for key establishment leveraging channel anonymity, submitted to IEEE Transactions on Information and Forensics Security (TIFS).
3. Savio Sciancalepore, Giuseppe Piro, Daniele Caldarola, Gennaro Boggia and Giuseppe Bianchi, OAuth-IoT: an access control framework for the Internet of Things based on open standards, submitted to the 22nd IEEE Symposium on Computers and Communications (ISCC 2017).

0. PERSONAL SCIENTIFIC CONTRIBUTIONS

Table 1: Overview of the scientific publications included in this PhD dissertation

Contribution	Chapter - Paragraph	Research Area	Layer of the Protocol Stack
Int. Conf. 4	3.3	Security	MAC
Int. Journ. 5	3.1	Security	MAC
Int. Conf. 3	3.4	Security	MAC
Int. Conf. 2	4.1	Security	MAC
Int. Journ. 4	5	Wireless Localization	PHY
Int. Journ. 3	3.2	Security	MAC
Int. Journ. 2	3.3	Security	MAC
Int. Journ. 1	4.1	Security	APP
Int. Conf. 1	4.2	Security	APP
Int. Journ. 1	5	Wireless Localization	PHY
Int. Journ. 2	3.5	Security	PHY
Int. Conf. 3	4.2	Security	APP

1

The Internet of Things technology

In this chapter the IoT technology is briefly introduced, along with the wide set of application domains that will benefit from its integration and development. The most important emerging standardized protocol stack for this technology is described in detail, along with the relationship of included protocols with state of the art approaches. A high-level view of commercial hardware and software products, running and implementing the described protocol stack and already available on the market is also provided.

1.1 The rise of the Internet of Things

The Internet of Things (IoT) is generally envisioned as a networked system of smart interacting objects (i.e., sensors, machines, vehicles, smart phones, tablets, etc.) [4, 5]. These devices are able to acquire different data (i.e., temperature, light, acceleration, pressure, and so on) from the surrounding environment and, at the same time, receive and process in real-time input messages from the outside big Internet network, in order to react to particular events.

The expected benefits provided by the rise of the IoT are significant in terms of economic value and improvement of the quality of life [6]. According to recent forecasts of the major leading companies of the sector [7, 8, 9], around 50 billions devices are expected by 2020 in the IoT, more than 6 times the world population, as depicted

1. THE INTERNET OF THINGS TECHNOLOGY

in Figure 1.1. This uncontrolled explosion of smart devices will pave the way to novel pervasive services in many application domains, such as: health care, smart city, energy management, military, environmental monitoring, and industry-automation, to name a few [10]. Moreover, this further motivates the current efforts by researchers, industries, and standardization bodies worldwide, devoted to solve the plethora of issues still open in this ebullient context [10, 11, 12].

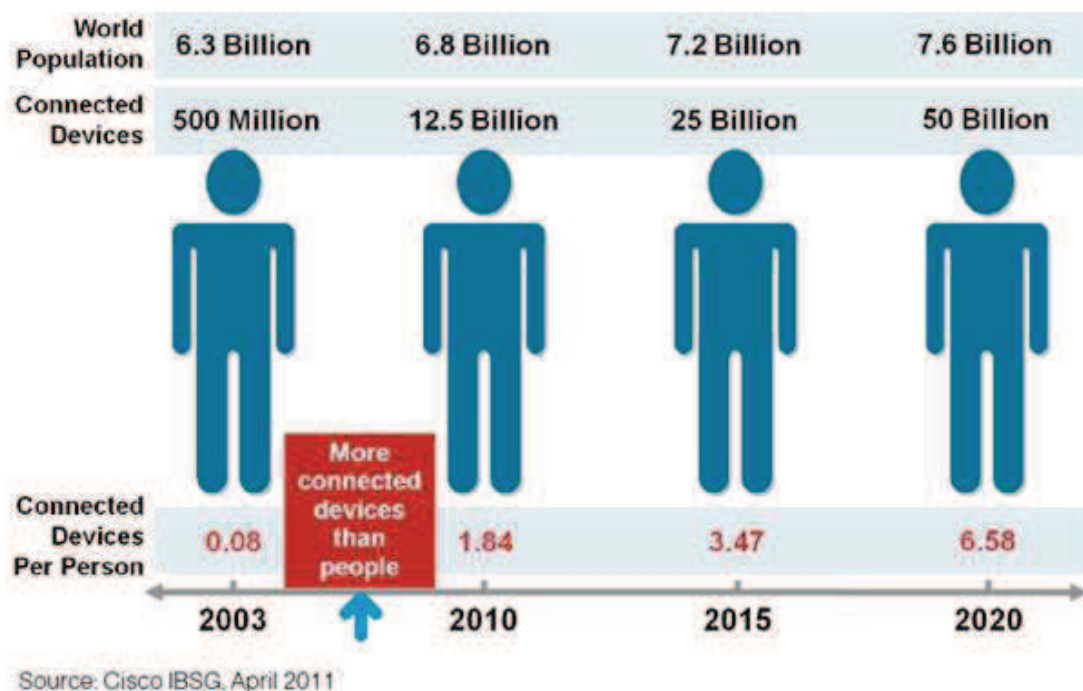


Figure 1.1: Evolution of number of IoT devices and world population in the last decade.

Smart objects that create and enable the IoT are generally constrained, in terms of computational, storing, and communication capabilities. They usually communicate over loosely-coupled short-range low-power wireless links up to a central storing and computational element, able to provide coordination capabilities for the entire network.

Since from its appearance in 1999 from Kevin Ashton [13], IoT attracted a lot of efforts. During the last decade its meaning and scope has undergone a lot of modifications and adjustments, from Wireless Sensor Networks (WSN) to Industrial IoT (IIoT). The current trend is evolving toward the concept of the Internet of Everything (IoE), i.e., the intelligent connection of people, process, data and things. The vision of the IoE describes a world where billions of objects are enriched with sensors, able to detect,

measure and assess their status in real-time. All such solutions use devices connected over public or private networks using standard and proprietary protocols. It is worth noting that IoE builds on top of the IoT, using things in a wider fashion, with improved connections with people, data and processes. These further advance the power of the IoT, with improved business opportunities and industry outcomes. Techniques that empower the IoT and bridge the gap toward the enabling of the IoE are further detailed in the next sections of this thesis.

1.2 Applications of IoT technology

With the advent of the IoT, billions of objects will be scattered almost everywhere to enable smart services in relevant application domains, such as health care, logistics, energy management, military, environmental monitoring, and industry-automation, to name a few [10]. A high-level view is depicted in Figure 1.2.

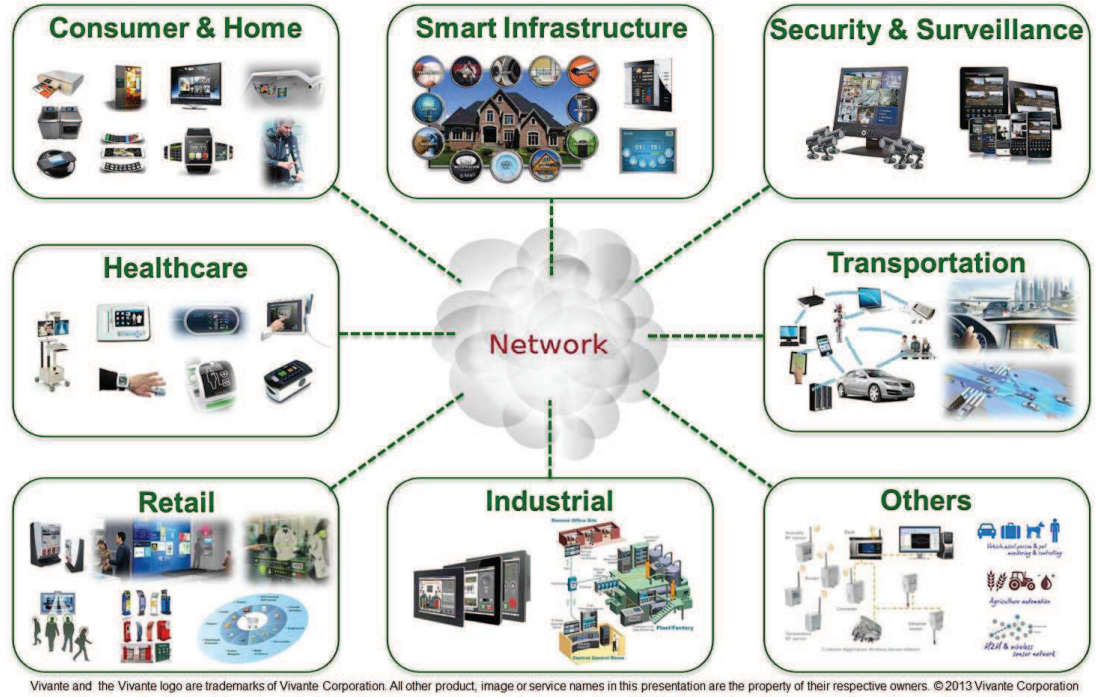


Figure 1.2: High-level view of IoT application domains.

With reference to health-care specific implications, the IoT technology can be of immense importance because of the capability to provide heterogeneous data acquired

1. THE INTERNET OF THINGS TECHNOLOGY

directly from the patient, without requiring him to be physically present, with enhanced efficiency and comprehensive knowledge. By introducing IoT technologies in healthcare, legacy investigation and treatment techniques can be augmented and integrated with advanced sensing capabilities, revolutionizing the relationship between doctors, health-care devices, and patients [14].

There are currently a number of IoT commercial solutions and technologies working in the eHealth application domain, providing data whose features were unavailable before and, at the same time, enabling care also to people previously unreachable for many reasons. IoT applications can improve the clinical care, by constantly monitoring patients in the hospital that effectively require constant supervision. Radical 7¹, provided by the *Masimo* company, is a state-of-the art example in such a context. It is a non-invasive measurement system, able to acquire data autonomously while giving to doctors the possibility to tune specific parameters on a patient basis. It integrates 802.11 and Bluetooth connectivity capabilities, and allows doctors to retrieve and analyse data both in real time from the measurement system or offline. At the same time the IoT technology can enable also remote monitoring of patients, by connecting wearable and implantable devices to the Internet and, consequently, directly with the hospital and the medical staff. The Home Health-Hub (HHH) reference platform² is a tangible product working in this direction. The HHH system represents a tele-health application, that integrates seamless connectivity and data aggregation to provide remote access to the patient and improved health-care management. Between the wide set of advanced features that it supports, it is possible to mention the capability to acquire vital sign measurements, the possibility to set up daily activity alarms and an advanced option that allows to consult anytime with the monitoring centre, medical staff, family and friends. The HHH system captures patient data from a multiplicity of sensors and smart devices, and securely stores it in the cloud, where medical staff can easily examine them. Finally, IoT-driven healthcare applications can have a significant impact also on active people, supporting them any time and monitoring if anything suddenly happens. The Sonamba system³, for example, is able to communicate constantly with

¹<http://www.masimo.com/rainbow/radical7.htm>

²http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/i.mx-applications-processors/i.mx28-processors/home-health-hub-reference-platform:RDHEALTH_HUB

³<http://sonamba.com/>

a set of sensors, tactically placed in the home environment, with the aim of monitoring daily activities of an individual and report anomalies to the medical staff. In the same scope, Libris technology ¹, provided by Numera, have been recently introduced in the market and it is able to detect individual falls both at home or away [15].

Significant efforts could be provided by the IoT technology also in the logistics. A connect pallet for example, can instantly and easily provide to its owner information about the current location and condition of its shipment. A connected truck can intelligently predict its own maintenance needs. A connected street light can sense the presence of cars and send environmental intelligence to drivers ².

Through the introduction of smart sensors and smart meters at the machine and production line level, the IoT paradigm promises to increase the visibility and awareness of energy consumption also. Real-time energy consumption data from manufacturing processes can be easily collected and then analyzed, to improve energy-aware decision-making [16]. The complete awareness about the required energy in a system could provide benefits both in terms of ecological and economic aspects, thus leading to an efficient management of the whole energy available in the system and the provision of energy only to the part of the system that effectively needs it.

In the military domain, potential application areas derive from the previous discussion, and include command and control of combined operations, fully immersive virtual simulations for soldiers' training, autonomous vehicles, situational awareness, target positioning, marking, vehicles, soldiers status and environmental monitoring, medical care (battlefield health monitoring, patients monitoring) and energy management ³.

Probably most famous applications of the IoT technology include the environmental and ecology area. Smart devices can be easily distributed in a wood or in a whatever area with the aim of monitoring critical parameters, such as temperature, pollution level, movement and radiation level, between the others. The exceeding of predefined threshold can easily trigger both automatic reactions (i.e., the activation of some cooling equipment) or notifications to human personnel, that can be instantaneously aware of the danger situation.

¹<http://numera.com/libris/>

²http://www.dhl.com/en/about_us/logistics_insights/dhl_trend_research/internet_of_things.html

³<http://www.fedtechmagazine.com/article/2016/05/internet-things-battlefield-needs-be-flexible-army-official-says>

1. THE INTERNET OF THINGS TECHNOLOGY

Between the areas in which IoT is gaining the highest level of attention by researchers and industry, the industry-automation domain is for sure the leader among the others. As highlighted in a recently published report by General Electric, the operational improvements due to real-time data collected from an IIoT yields Billions of Dollars in return of investment [17]. Possible applications of such an IIoT include telemetry readings about the status of oil/brakes/etc of cars on the move, health state measurements of blood pressure/heartbeat of elderly, monitoring of corrosion state of oil/gas pipelines, data about the number of empty locations in parking within cities, remote metering of water consumption, real-time monitoring of critical parts of a machinery, to name a few.

IIoT incorporates machine learning technique and big data technology, integrating seamlessly sensor data, Machine-to-Machine (M2M) communication and automation technologies that have existed in industrial settings for years. Gathered data can enable companies to pick up on inefficiencies and problems promptly, saving time and money and supporting business intelligence efforts. In manufacturing specifically, IIoT holds great potential for quality control, sustainable and green practices, supply chain traceability and overall supply chain efficiency.

The view that aims to integrate all these applications in a smart industry, able to serve the needing of the new century and to quickly adapt to highly dynamic markets is often referred to Industry 4.0 or as the forth industrial revolution.

1.3 Toward the definition of a Standard Protocol Stacks for IoT communications

Even if applications are ample, the engineering efforts needed to enable the aforementioned innovative services are tremendous. Without any doubt, however, only a rigorously standardized approach at data transmission, data handling, servicing and policy level will facilitate a viable technology up-take [17].

With this target in mind, in 2003, the IEEE started its work aimed to federate the communication protocol of the emerging low power wireless sensors and actuator systems. As an outcome, a lot of public and private agencies started to work on customized and/or open-source protocol stacks, developed for specific application areas.

1.3 Toward the definition of a Standard Protocol Stacks for IoT communications

In the following few significant examples are provided to show the evolution of protocol stacks for IoT devices towards energy-efficiency, security and advanced optimization techniques.

1.3.1 ISA 100.11a and WirelessHART specifications

In the early 2000's, research activities in the topic of efficient wireless communication stacks in WSN and industrial environments produced a great variety of technologies and proprietary solutions, each of them focused on a given context. Between these, WirelessHART and ISA 100.11a are the most significant. In fact, they firstly introduced MAC protocols with a rationale very close to the Time Synchronized Channel Hopping (TSCH) protocol, today widely recognized as one the most successful techniques to enable reliable IoT communications.

The following basic components can be identified within a WirelessHART network:

- *Field Device*: it is a common device, equipped with wireless communication capabilities;
- *Adapter*: it is a component that can be plugged within a field device in order to enable WirelessHART communication capabilities;
- *Handheld*: it is a portable device, able to communicate using the WirelessHART technology;
- *Gateway*: it is an access point that connects the WirelessHART network to the plant automation network;
- *Network Manager*: a component responsible for the management of the whole WirelessHART network;
- *Security Manager*: the component who is in charge of managing the cryptographic materials (i.e., session keys) [18].

Typically, the gateway, network manager, and security manager reside in the same embedded device, usually referred to as either the gateway or network manager by the vendors [19].

Similar roles can be identified within an ISA100.11a installation, as follows [20]:

1. THE INTERNET OF THINGS TECHNOLOGY

- *Input/Output (I/O)*: an end-leaf device, providing or receiving data (sensor or actuator, respectively).
- *Router*: device able to switch and route messages from other devices in the ISA100.11a installation;
- *Provisioning*: a device which is in charge of making possible for other devices the join to an established ISA100.11a network;
- *Backbone Router*: border device able to manage data coming in or out from the network;
- *Gateway*: it is an interface between the wireless and the cabled network;
- *System Manager*: a component responsible for the management of the network;
- *Security Manager*: a component that enables the security in the network, in strict cooperation with the system manager.
- *System Time Source*: a component responsible for time alignment within the network.

Apart from the differences that it is possible to notice at the field layer, the importance of both WirelessHART and ISA100.11a grounds on the introduction of the TSCH at the MAC layer. TSCH was part of IEEE802.15.4e standard [21], and now it has been merged with the official release, being included in the recent IEEE802.15.4-2015 [22]. It is based on a trivial but ingenious idea: synchronize the nodes for obtaining energy efficiency. In parallel, the nodes apply channel hopping. The first minimizes contention, while the second increases reliability since gaining in frequency diversity [17].

WirelessHART technology orchestrates all communications on a Time Division Multiple Access (TDMA) scheme, and it adopts a channel hopping scheme in order to increase data bandwidth and robustness to interferences and packet losses.

The WirelessHART specification defines a 10 ms time-slot and includes a single algorithm to obtain the channel hopping sequence. The most used topologies are meshed, and traffic is routed through graph routes, in which one or more path coexist at the same time. The *Network Manager* centrally performs the scheduling of the communications in the network, by looking at requirements specified by field devices and applications.

1.3 Toward the definition of a Standard Protocol Stacks for IoT communications

The Data Link Layer (DLL) used in ISA100.11a leverages a non-compliant form of the IEEE802.15.4 MAC. From one side, as for the WirelessHART DLL, it includes frequency hopping and time slotted TDMA features. From the other side, it allows for an higher flexibility, by allowing a configurable time-slot size and the possibility to choose between three channel hopping methods (i.e., slotted, slow, and hybrid) with five channel hopping patterns, are supported. The *System Manager* creates, updates and manages graphs used to route the data in the network, according to throughput and transmission requirements defined by the network designer.

Despite the introduction of these interesting functionalities, WirelessHART and ISA100.11a still remain industrial protocols, designed to be application-specific technologies.

1.3.2 Zigbee protocol stack

A step ahead toward the standardization process was performed through the foundation of the Zigbee Alliance in 2002¹.

The architecture of the Zigbee protocol stack is divided in different layers. A high-level view is showed in Figure 1.3².

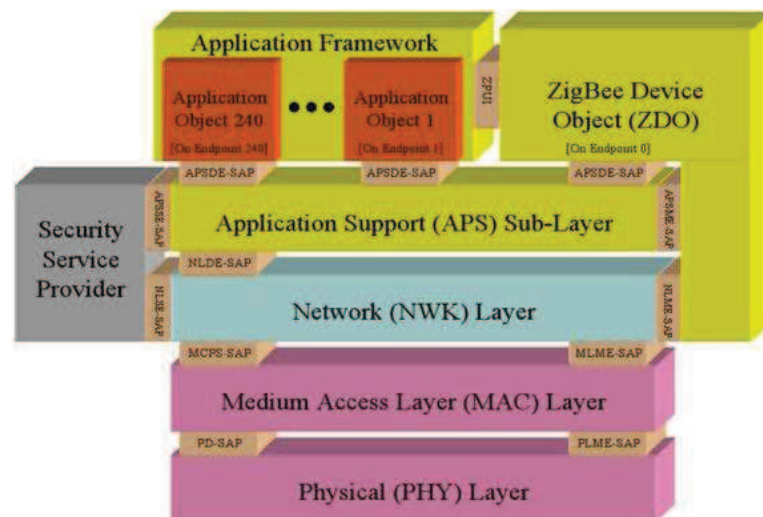


Figure 1.3: High-level view of the Zigbee Protocol Stack.

¹www.zigbee.org

²Zigbee Open House - Seoul, 2006 - Phil Jamieson

1. THE INTERNET OF THINGS TECHNOLOGY

The two lowest layers of the stack are derived from the IEEE 802.15.4-2003 standard. The ZigBee Alliance builds on these foundations by providing the NetWork Layer (NWK) layer and the framework for the application layer [23].

The application layer framework is split in two different sub-layers: the APplication Support sub-layer (APS) and the Application framework, that are connected each other thanks to the Zigbee Device Object (ZDO). The APS provides interfaces and services used by upper-layers to interface with the NWK, for data managing purposes.

The Application Framework provides two types of data services. From one side, it creates and manages *Generic messages*, defined as a developer customized structure. From the other side, it manages *key-value* pairs, used for getting attributes within the application objects.

Between these two sub-layers, the ZDO provides an interface between them. Specifically, it is in charge of detecting, initiating and binding other devices to the network.

Manufacturer-defined application objects use the framework and share APS and security services with the ZDO.

1.3.2.1 IEEE 802.15.4-2003

It is worth going more in detail regarding the IEEE 802.15.4-2003 standard specification, because it represents the basis of the technology over which this thesis have been built. It allows the possibility to choose one of the two PHY layer defined herein, operating in the ranges 868/915 MHz and 2.4 GHz (this twofold choice is due to the possibility that one of the two cannot be used because of local legislation issues).

The MAC sub-layer defined within the IEEE 802.15.4-2003 standard controls access to the radio channel using a pure Carrier Sense Multiple Access (CSMA) - Collision Avoidance (CSMA-CA) mechanism, illustrated in Figure 1.4 [24]:

The CSMA-CA algorithm is executed every time a node has to transmit a data frame.

A device running the IEEE802.15.4-2003 MAC maintains three variables for each transmission attempt:

- *NB*, is the number of times the CSMA-CA goes into back-off procedure;
- *CW*, is the length of the contention window;

1.3 Toward the definition of a Standard Protocol Stacks for IoT communications

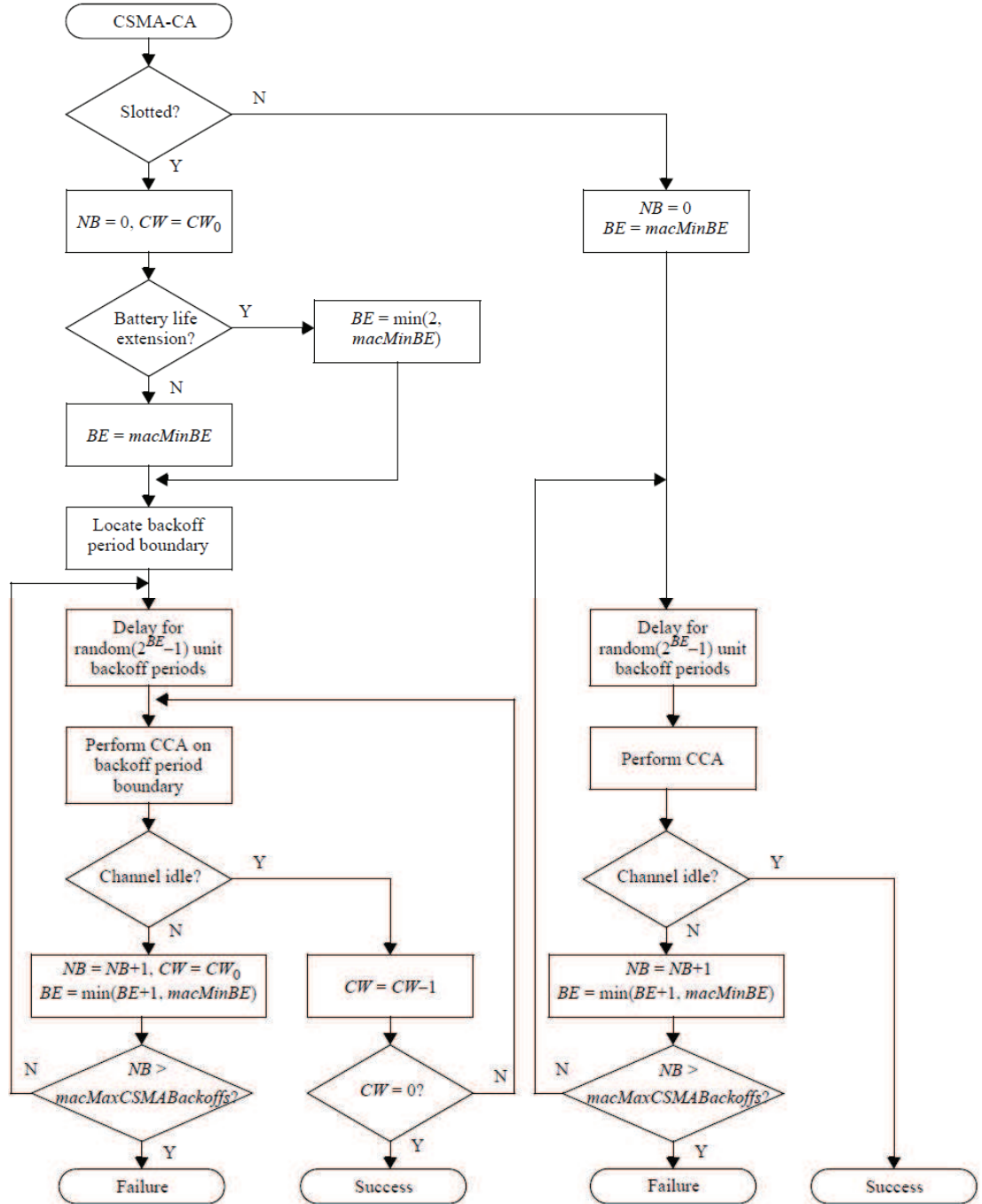


Figure 1.4: Flow diagram of the CSMA-CA protocol.

1. THE INTERNET OF THINGS TECHNOLOGY

- BE , that is the backoff exponent, directly related to the number of backoff periods that a device must wait before attempting to access to the channel.

Apart from these implementation-specific variables, the algorithm runs as a standard CSMA-CA protocol. In the slotted mode, the device locates the boundary of the slot and then it extracts a random backoff value. At the expiration of this time, it listens for the channel, assessing if it is clear or not. If it is clear, it tries to transmit; if it is already busy, it extracts another backoff period and re-schedule the transmission, until a maximum number of attempts.

The responsibilities of the IEEE802.15.4 technology includes also transmitting beacon frames, synchronization, and providing a secure transmission mechanism.

The channel access is regulated by a specific *Super Frame* structure, as illustrated in Figure 1.5, composed by 16 consecutive slots (all with the same duration). In order to communicate some details about the PAN and the medium utilization, as well as enabling synchronization procedures for all the attached devices, the coordinator periodically releases a control message, namely *Beacon*, at the beginning of the *Super Frame*. Moreover, as depicted in Figure 1.5, the transmission of messages is done during the *active period* (i.e., a first part of the *Super Frame*). The remaining unused slots form the inactive period, during which devices can turn in low-power mode for energy-saving purposes [25].

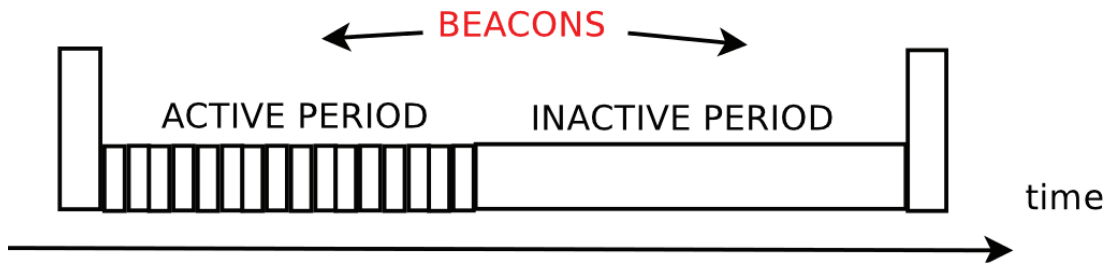


Figure 1.5: The SuperFrame Structure.

1.3.3 ZigBee IP

Despite its revolutionary features and its enormous impact on the scientific and industrial community, the legacy ZigBee specification does not allow an easy plug of

1.3 Toward the definition of a Standard Protocol Stacks for IoT communications

distributed WSN into the IP-based Internet. Therefore, the Internet Engineering Task Force (IETF) standardization body developed a new set of protocols, supporting communications based on the IPv6 protocol even in IoT-compliant multi-hop Low-power and Lossy Networks (LLNs). The protocols proposed by the IPv6 over LLN (6LoWPAN), Routing Over LLN (ROLL) and Constrained RESTful Environments (CoRE) IETF Working Groups (WGs), along with UDP and IPv6, have been integrated to create the ZigBee IP protocol stack, pictured in Figure 1.6 [26].

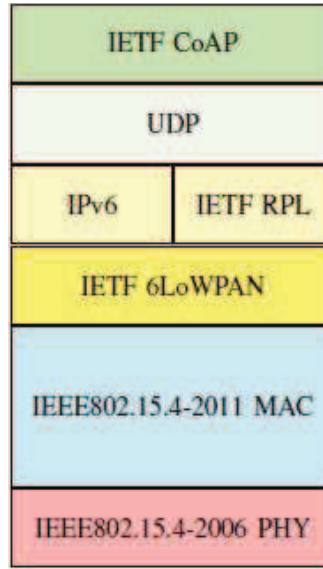


Figure 1.6: ZigBee-IP protocol stack.

1.3.4 Bluetooth technology

Bluetooth is a radio-frequency technology, originally designed by Ericsson, that works in the unlicensed 2.4GHz ISM band. It is a short-range low-power technology, whose transmission area can vary between 1 meter and 100 meters. Currently, there exist many different Bluetooth specifications, defined by the Bluetooth Special Interest Group (SIG), composed by over 30.000 companies worldwide [27].

Bluetooth standard specifications have been designed to support compatibility with previous versions, while improving some specific features. The last is Bluetooth v5, announced in June 2016, which increases the transmission range, the speed and bandwidth with respect to previous specification, with a main focus on IoT-based devices

1. THE INTERNET OF THINGS TECHNOLOGY

and services.

The figure 1.7 provides a big picture of the Bluetooth protocol stack.

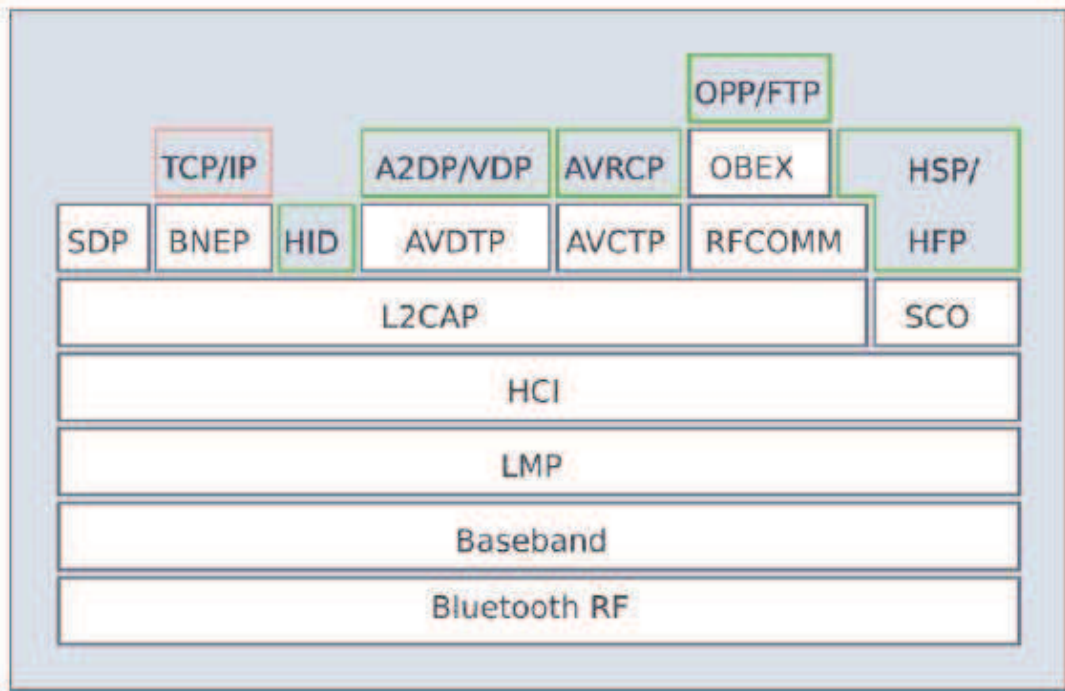


Figure 1.7: Big picture of the Bluetooth protocol stack.

Starting from the bottom, the first three layers are often indicated as the Controller Part. They include the radio (that can be Basic Rate, Low Energy or High-Speed, depending on the given specification), and it is usually confined to the hardware. The upper part, also called the Host part, mostly depends on the chosen specification, and includes a set of proprietary protocols dedicated to specific tasks.

Nowadays, Bluetooth is probably the most famous and used technology in the context of short-range low-power solutions. In fact, its uses cover wireless headsets, keyboards and mice, smart PCs, smart cameras, low-rate data exchange and also automatic messages.

Moreover, it fully supports physical layer robust techniques, such as the Frequency Hopping Spread Spectrum (FHSS) scheme. However, the Bluetooth technology only supports star topologies, in which devices are directly connected to the sink of the network.

1.3.5 6tisch protocol stack

In parallel with the widespread of the ZigBee IP technology, issues related to the low reliability and complete unpredictability of the IEEE 802.15.4 MAC clearly emerged. These were due essentially to the radio overhearing and idle-listening included in the CSMA-CA algorithm, which resulted in an high interference level especially in crowded networks, challenging industrial environments and multi-hop/meshed topologies [28].

Starting from the efforts produced within the WirelessHART technology, and the outstanding performances of the Time Synchronized Mesh Protocol (TSMP) in industrial applications, in 2011 the IEEE standardization body integrated the foundation of TSMP, i.e., the TSCH protocol, into the IEEE802.15.4e-2012 MAC amendment [21].

IEEE802.15.4e guarantees an high-level of flexibility, efficiency and reliability, paving the way to the adoption of the IEEE802.15.4 technology also in challenging scenarios, such as industrial applications and critical communications. However, its integration with the highest layers of the protocol stack is not immediate, especially for what concern initialization and managing of security and scheduling features.

Therefore, a new IETF WG has been formed, namely IPv6 over the TSCH mode of the IEEE 802.15.4 (6tisch), whose aim is to setup an intermediate layer able to manage and complete the integration between IPv6 and the TSCH mode of the IEEE802.15.4e-2012 standard. The resulting protocol stack designed within this emerging IETF WG is depicted in Figure 1.8 and described in what follows.

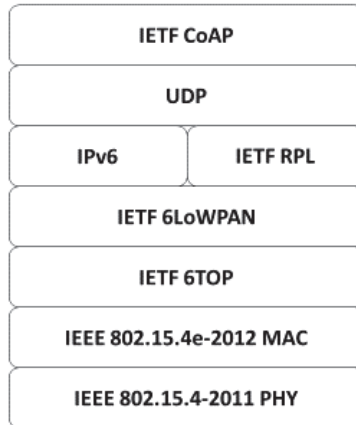


Figure 1.8: Communication stack integrated in the *TLSensing* platform.

As the most up-to-date and promising technology at the time of this writing, the

1. THE INTERNET OF THINGS TECHNOLOGY

6tisch stack will be the reference protocol suite for all discussions carried out in this thesis.

1.3.5.1 Physical layer: IEEE 802.15.4-2006 standard

In general, the role of a Physical (PHY) layer is to determine the radio frequency used for radio operation, the modulation format and the encoding of the signal.

The IEEE802.15.4-2006 standard defined both the PHY and the Media Access Control (MAC) layer, even if this latter one has been further modified by later amendments.

The IEEE 802.15.4 PHY is a healthy trade-off between energy-efficiency, range, and data rate targeted at building-sized networks [28]. As for the version published in 2003, it defines two different PHY layers, even if the the most widely used is the one operating in the unlicensed frequency band ranging from 2.4 GHz to 2.485 GHz. In this frequency band, the standard defines 16 orthogonal channels, located every 5 MHz and 2 MHz wide. The maximum time necessary to switch from a channel to another is $192\mu\text{s}$.

In this band, the IEEE 802.15.4 PHY layer uses the Offset-Quadrature Phase-Shift Keying (O-QPSK) modulation scheme, with a 2 Mbps physical data rate. Looking deeply into a IEEE802.15.4-compliant radio, during the transmission phase every group of 4 bits is encoded as 32 chips (i.e., physical bits) by following standard-defined lookup tables. The maximum bit-rate that can be reached from a user's perspective is 250 kbps, even if internally further 8 more chips are delivered a 2 Mcps link. This technique, also known as Direct Sequence Spread Spectrum (DSSS), is a well-known scheme that provides further robustness to the delivered radio signal.

From a protocol perspective, before initiating the transmission of a packet the radio transmits a physical preamble $128\mu\text{s}$ long to allow the receiver to lock to the signal. Then, the Start of Frame Delimiter (SFD) well-known sequence is transmitted to indicate the start of the PHY payload. The first byte of this payload is dedicated to the indication of the length of the payload itself. The IEEE802.15.4 technology defines a Maximum Transmission Unit (MTU) of 128 bytes at the PHY layer, including the above-mentioned length byte. This limit is often a physical limit of the radio, that dedicates a portion of memory exactly equal to 128 bytes to buffer the received signals and the packets that needs to be delivered over the wireless channel.

1.3 Toward the definition of a Standard Protocol Stacks for IoT communications

Finally, note that each radio packet is closed by an End Of Frame delimiter (EOF) sequence, used to communicate the end of the PHY payload.

1.3.5.2 MAC layer: IEEE 802.15.4e-2012 amendment

The IEEE802.15.4-2006 standard, directly evolving from the previous IEEE802.15.4-2003 specification, also defines a MAC protocol based on the CSMA-CA algorithm, as described in Section 1.3.2.

However, this MAC layer offers good performances only in star networks, in which all nodes communicate directly through a central coordinator mote. When tested over low-power multi-hop networking topologies, it fails mainly because of the following reasons:

- Energy inefficiency for sink nodes: the IEEE802.15.4 legacy MAC does not explicitly forbid multi-hop mesh topologies, but requires the coordinator or, generally speaking, the potential receiver, to have its radio always on listening for incoming messages. This can be tolerated in a star topology, where the central node uses to be mains-supplied, but it becomes completely unfeasible when the receiver is battery-powered, as in a multi-hop topology.
- Single-channel operation: the IEEE802.15.4 legacy MAC forces the nodes to operate in a single channel in the set defined in the previous subsection, without any interleaving technique. However, the wireless channel is unpredictable and unreliable in its nature, and the above constraint results always in a low level of reliability and packet delivery.

The exploit of the IEEE802.15.4e is due essentially to the resolution of these two main problems [21].

Even if this specification addresses just the MAC layer, thanks to time synchronization and channel hopping, it enables high reliability while maintaining very low duty cycles, both highly recommended requirements for the emerging IoT [28]. Specifically, the synchronization between the involved devices provides energy efficiency, while the channel hopping scheme improves the reliability of the communication.

As in IEEE 802.15.4, two types of network nodes, i.e., the Fully Functioned Device (FFD) and the Reduced Functioned Device (RFD), can be found in the network. They

1. THE INTERNET OF THINGS TECHNOLOGY

could be arranged in both peer-to-peer and star topologies. A FFD has the highest computational capabilities and it usually works as the coordinator of the network (also called PAN coordinator), thus being a reference node for a group of others RFD devices. Instead, a RFD has lower resource and communication capabilities and it is able to communicate only with its reference FFD. Whereas RFDs must be connected only to a single coordinator, FFDs can connect among them forming more complex meshed network architectures [25].

Differently from IEEE 802.15.4, in the TSCH mode defined by IEEE 802.15.4e the nodes synchronize on a slotframe structure. With the term *slotframe* it is possible to address a sequence of time-slots, that repeats over time. The Absolute Slot Number (ASN) counts the total number of slots that elapse from the beginning of the time, i.e. from the turn on of the coordinator node of the network. In general, in a slot a device can transmit, receive or maintaining the radio off, thus performing a TX, RX or SLEEP slot, respectively. The time duration of a slot is a very important parameter in a network operating in the TSCH mode. In fact, a slot must be long enough for the sender to transmit a frame and receive the acknowledgment for this transmission by the receiver. This is a very important concept, that will be recalled also in the following chapters of this thesis. The standard defines a minimum slot duration of 10 ms, even if different choices are allowed. This multiple option is due to the fact that there are some class of devices that are unable to perform the above operation in 10 ms. However, IEEE 802.15.4 must run also on these devices.

Moreover, to maintain devices' synchronization in each slot, IEEE 802.15.4e specifies two possible methods: frame-based or ack-based synchronization. In both of the cases, the receiver calculates the difference between the expected time of frame arrival of a frame from its parent node and its actual arrival. The difference between the two techniques consists in the fact that in the frame-based scheme the receiver tune its clock using this difference, while in the ack-based synchronization approach the information about the difference is delivered to the sender in the ack and the sender adjusts its clock by the difference. Independently from the chosen technique, the direct communication is the fundamental building block for maintaining synchronization. Therefore, when a too long *silence* phase occurs, the nodes exchange a keep-alive packet to verify the presence of the neighbor.

1.3 Toward the definition of a Standard Protocol Stacks for IoT communications

If the slot-frame structure provides deterministic access to the wireless medium and benefits in terms of energy consumption, the channel hopping strategy provides reliable communications, by mitigating the effects of interference and multi-path fading.

Defining as $(t, chOf)$ the specific slot and the channel offset, ASN as the ASN, and n_{ch} as the number of available channels (16 in IEEE 802.15.4 technology), the frequency (i.e., a real channel) used for the transmission of a packet is calculated by using the following Equation 1.1.

$$f = F\{f(ASN + chOf) \bmod n_{ch}\}. \quad (1.1)$$

Even if IEEE 802.15.4 defines 16 channels, each specific deployment can use a blacklist can to restrict the deny the use of some channels, typically for coexistence purposes.

IEEE 802.15.4e defines the operations performed by each node in a slot, given the nature of the slot, i.e., TX, RX or sleep. However, it does not specify how nodes can agree on the nature of a slot, i.e. how they build the schedule. This is an aspect of key importance, because when a node transmits to another node in a slot, that receiving node must have its radio on in order to receive the message. Moreover, given the nature of the networks, nodes come and go frequently, as well as are refreshed constantly. In general, scheduling techniques can be classified in centralized distributed. In a centralized approach, there is a central entity, typically the PAN Coordinator, that is responsible for building and maintaining the schedule. Differently, a distributed approach involves the nodes to decide locally on which links to schedule with which neighbors.

Note that the higher the slot duration, the lower the throughput of the network and, in parallel, the lower the energy consumed in a given period of time (comparing two schedules arranged in the same way).

1.3.5.3 6top and 6tisch-related activities

As previously anticipated, IEEE 802.15.4e defines how nodes behave in a given slot. However, it does not defined how a schedule can be initiated, negotiated among peers and maintained. To bridge this gap, the IETF 6tisch WG introduced the 6tisch Operation Sublayer (6top) [29] [30]. 6top is the next higher layer of the IEEE802.15.4 TSCH

1. THE INTERNET OF THINGS TECHNOLOGY

medium access control layer, and it is globally responsible for initiating, configuring and maintaining a network based on the TSCH mode of the IEEE 802.15.4 technology.

As part of the 6top layer, the 6top Protocol (6P) enables a distributed scheduling, while allowing neighbor nodes to add/delete TSCH cells to one another. A *cell* is a specific communication channel, defined as a combination of a time and a frequency channel, that is assigned to a given couple of nodes.

In the 6tisch architecture, there is a central entity, namely the Path Computation Element (PCE) that centrally manages the whole network. By using the PCE Protocol (PCEP) protocol, it communicates with a Path Computation Client (PCC) to establish a communication channel.

The 6tisch WG is working on a twofold set of scheduling techniques, by allowing both centralized and distributed approach. Towards the centralized logic, 6tisch is defining mechanisms through which a PCE can manage the schedule through Constrained Application Protocol (CoAP) communications, given that the PCEP works on a TCP established connection. Moreover, the 6tisch WG is focusing on defining a specific network architecture that provides low-latency, low-jitter and high-reliability packet delivery [31].

In the distributed direction, instead, 6tisch is working on the On The Fly scheduling (OTF) protocol, that is able to dynamically allocate cells between a couple of neighbors, while requiring minimal bandwidth and avoiding collisions [26].

Given the availability of some MAC extensions in the IEEE802.15.4 frame header format, the 6tisch WG is defining new Information Elements. These pieces of information can be exchanged between single-hop neighbors or forwarded up to farther devices, allowing for several optimizations.

The 6top layer cares also about MAC-layer security. In fact, as will be deeply discussed in the following, the IEEE 802.15.4 standard defines mechanisms through which a packet can be delivered securely and correctly received. However, it does not specify how the key can be negotiated among peers, maintained and updated. For these reasons, the 6tisch WG is working on defining techniques for new nodes to securely join an already established 6tisch network, and for the PCE to provide dedicated session keys for communication between constrained devices [32].

Finally, the 6tisch WG is defining a minimal mode of operation for a 6tisch network, in order to provide IPv6 connectivity over a constrained network, which consists of

IEEE 802.15.4 TSCH links. In this sense, the reference document [33] identifies and integrates a collection of protocols, to enable interoperable IPv6 connectivity over IEEE 802.15.4 TSCH with minimal network configuration and infrastructure.

1.3.5.4 6LoWPAN Adaptation layer

The work of the 6LoWPAN WG [34] has been dedicated to the standardization of the 6LoWPAN protocol [35].

It is an adaptation layer, able to effectively plug low power technologies (such as IEEE802.15.4) to the Internet. The adoption of the IPv6 protocol on top of a LLN is not straightforward, and requires a lot of optimizations. First, the MTU defined by the IEEE802.15.4 physical layer, that is 127 bytes at the MAC layer, imposes an optimization of the IPv6 header and the format of the IPv6 address, in order to allow a packet not to be fragmented every time (note that the default MTU of the IPv6 protocol is 1280 bytes and the header is 40 bytes long). 6LoWPAN solved a number of issues, i.e., handling network auto-configuration, auto-configuration of IPv6 addresses, supporting applications with a high number of devices (and of addresses), easing the internet-working with other IP-based infrastructures, enabling link-layer forwarding and fragmentation, compressing IPv6 headers and Next Headers (by suppressing redundant information that can be inferred from other layers in the communication stack), reduction of routing and management overhead, adoption of lightweight application protocols (or novel data encoding techniques), and the support for security mechanisms [26, 36].

1.3.5.5 Network layer: RPL protocol

The Routing Protocol for LLN (RPL) protocol has been produced by the ROLL WG, and it has been standardized in RFC 6550 [37].

It is a gradient-based distance-vector routing protocol, that eases the formation and the management of LLN. One of its most important feature is the capability to run under a variety of link layer protocols, independently from their reliability. Furthermore, it has been designed to be strictly adherent to the IPv6 protocol: in fact, it uses signaling messages carried as options of IPv6 Router Advertisements.

Typically, RPL protocol organizes the routing through the establishment of a small set of sinks, that gathers the routing traffic and routes the message. In details, RPL

1. THE INTERNET OF THINGS TECHNOLOGY

organizes the network topology as a Directed Acyclic Graph (DAG), further partitioned into one or more Destination-Oriented DAGs (DODAGs), each of them having a root that coincides with a sink. A DAG can be created according to different metrics, depending on the type of installation or the choose of the network administrator: available energy resources, workload, throughput, latency, reliability of the link, to name a few. To this aim, RPL uses an Objective Function, defined in many different ways to allow for enhanced flexibility, independently from the particular deployment scenario. The construction of the topology is based on the exchange of two types of messages: DODAG Information Object (DIO) and Destination Advertisement Object (DAO). DIO are delivered to establish paths towards the roots, and contain information about the position of the sender with respect to the sink (i.e., the rank), the objective function, IDs, and so forth; DAO, instead, are used to establish inverse paths toward the leaf nodes of the DAG. Moreover, RPL addresses also security aspects at the networking layer by defining three different security modes (unsecured, pre-installed and authenticated), to be chosen according to specific needings and threads of the network. As for IEEE802.15.4 technologies, the confidentiality is managed through the AES-128 encryption scheme.

1.3.5.6 Transport layer: UDP Protocol in IoT technology

The Transport layer is responsible of providing end-to-end reliability over IP based networks. Even if the Transmission Control Protocol (TCP) provides advanced features such as traffic and congestion control, the amount of packets needed to integrate such features and a reliable transmission would not be sustainable by a LLN, in terms of number of transmitted packets and end to end packet confirmation which directly maps to energy consumption [28]. For all this reason, the User Datagram Protocol (UDP) [38] protocol has been widely recognized as the most suitable solution for the transport layer of an IoT network, while delegating features such as retransmission control mechanisms at the application layer.

As well known, UDP is a datagram oriented protocol that provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism and overhead. As highlighted in previous sections, the 6LoWPAN protocol works also on the transport layer to remove some fields not only in the IPv6 protocol

but also regarding UDP. In fact, these fields or take well-known values, or they can be somehow inferred from other fields in the IEEE802.15.4 header.

1.3.5.7 Application layer: the CoAP protocol

According to the IoT vision, constrained devices should directly communicate (by sending messages or being queried) through the public Internet according to the RESTful architecture and the client/server model. However, a straightforward implementation of the legacy HTTP protocol would be impossible, because of both memory and processing capabilities.

In order to enable interoperability while respecting the limits and the constraints of LLN devices, an adaptation protocol, namely CoAP, has been developed by the CoRE WG [39].

The CoAP protocol easily translates to HTTP for integration with the web. Moreover, it meets specialized requirements such as: multicast support, very low overhead, and simplicity for constrained environments.

By using CoAP, a subset of the HTTP operations can still be used when interacting with constrained devices. However, CoAP is an asynchronous request-response protocol, that runs over UDP at the transport layer, thus making substantially different with respect to the more famous HTTP.

From an high-level perspective, CoAP can be described as divided in two sub-layers: the message layer and the request/response layer. The former guarantees reliability and messages in-sequence arrival, while the latter associates requests to responses and the relative semantics.

As an advanced feature, CoAP allows the enabling of the *Observe* flag, that translates its features in a publish-subscribe paradigm. The enabling of the observe option allows for asynchronous notification sent to the clients by the server hosting a given resource. This allow to close the gap between this protocol and the more famous Message Queuing Telemetry Transport (MQTT) solution.

From the security perspective, CoAP recommends the integration with Datagram Transport Layer Security (DTLS) protocol. This feature will be detailed in the following sections, as well as the implications of such recommendations and the features that can be improved in this context.

1.4 Real IoT products

1.4.1 Hardware Devices

In the last years a lot of hardware devices have invaded the market, with the aim of enabling testing and deployment of standardizing protocol stack and customized solutions for the IoT. These devices are referred as *mote*, in order both to distinguish the terminology used for the hardware device with respect to its role in the network (the term used in this case will be *node*) and to identify the board as a whole (including processing and sensing capabilities). The evolution of such devices is following the current trends in the electronics and micro-electronics integration. In the following subsections three representative experimental boards will be introduced: TelosB (Section 1.4.1.1), Z1 (Section 1.4.1.2) and OpenMote-CC2538 (Section 1.4.1.3). These boards have been used for implementing and assessing performances of solutions designed in the following chapters. However, these are only a subset of the available commercial solutions¹.

1.4.1.1 TelosB

The TelosB mote, depicted in Figure 1.9, [40] (designed in 2004) features an MSP430 microcontroller (16-bit architecture, 8 MHz maximum CPU speed, 48 kByte flash, 10 kByte RAM), and a CC2420 radio chip [5].



Figure 1.9: TelosB board.

It has been designed by Memsic (old Crossbow) and it is considered, nowadays, as one of the most constrained boards available on the IoT market.

It includes two sensors: a SHT11 temperature sensor (which provides also humidity values) and a HamamatsuS1087 brightness sensor (absolute and relative) values). Moreover, it includes three leds (blue, red and yellow).

¹<http://www.postscapes.com/internet-of-things-hardware/>

1.4.1.2 Zolertia Z1

The Z1 module (see Figure 1.10) is a general purpose development platform for IoT, designed for researchers and developers [41]. The platform is compatible with the *Tmote*-family motes, as the TelosB described in Section 1.4.1.1, but includes several enhancements that offers roughly doubled performances in several aspects.



Figure 1.10: Z1 board.

It is equipped with two on board digital sensors (temperature and acceleration), 16-bit architecture, 16 MHz maximum CPU speed, 128 kByte flash, 8 kByte RAM, and a CC2420 radio chip.

It includes three integrated sensors: temperature (which provides also relative humidity values), light and battery charge.

Differently from the TelosB, on-board sensors are connected to the MSP430 processor through I2C instead of SPI bus.

Finally, it includes a 52-pin GPIO expansion connector, which allows to connect to a variety of phidgets sensors and relais.

1.4.1.3 OpenMote-CC2538

The OpenMote-CC2538 mote [42] (designed in 2014), depicted in Figure 1.11, is a state-of-the-art platform at the time to writing, which features a single-chip CC2538 solution [5].



Figure 1.11: OpenMote-CC2538 board.

1. THE INTERNET OF THINGS TECHNOLOGY

This chip contains both an ARM Cortex-M3 micro-controller (32-bit architecture, 32 MHz maximum CPU speed, 512 kByte flash, 32 kByte RAM) and a IEEE802.15.4-compliant radio.

The OpenMote family consists of three components:

- OpenMote-CC2538, is the portion of the constrained device that integrates computational capabilities;
- OpenBase, used for configuring both firmware and software of the aforementioned element;
- OpenBattery, includes four types of sensors, that measure temperature, relative humidity, ambient light, and acceleration. Their most important characteristics have been summarized in Table 1.1.

One immediate advantage of such a single-chip solution is that the micro-controller and radio chip share the same RAM memory; no time needs to be spent transferring data to/from the radio chip.

Nowadays, the OpenMote-CC2538 platform is considered as one of the most successful hardware device for the IoT.

1.4.2 Protocol stacks for the IoT

The lightweight protocol suite defined by IETF and detailed in Section 1.3.5 has already been implemented in many operating systems, such as OpenWSN, Contiki and RIOT, and it is ready to be used in real deployments [43].

In the following a big picture of the available frameworks and operating systems for IoT deployments will be provided.

1.4.2.1 OpenWSN

The OpenWSN project¹ proposes a free and open-source implementation of a standard-based protocol stack for constrained devices. It integrates several IEEE and IETF protocols, such as TSCH, CoAP, RPL and 6LoWPAN, and supports the creation of ultra-low power and highly reliable mesh networks connected among them through Internet [44]. Thanks to the integration of debugging and additional tools, it is able

¹<https://github.com/openwsn-berkeley>

Table 1.1: Sensors already available on the OpenBattery.

Sensor	Model	Manufacturer	Main characteristics
Relative humidity	SHT21	Sensirion	Resolution: 0.04 %RH @ 12 bits; 0.7 %RH @ 8bits Accuracy tolerance: ± 2 %RH (typical); ± 3 %RH (maximum) Response Time: 8 s Long Term Drift: < 0.25 %RH/years
Temperature	SHT21	Sensirion	Resolution: 0.01°C @ 14 bits; 0.04°C @ 12bits Accuracy Tolerance: $\pm 0.3^{\circ}\text{C}$ (typical); $\pm 1.6^{\circ}\text{C}$ (maximum) Operating Range: from -40°C to 125°C Response Time: from 5 s to 30 s Long Term Drift: $< 0.02^{\circ}\text{C}/\text{years}$
Light sensor	AX44009	Maxim	Maximum Lux Sensitivity: 0.045 Lux/LSB Saturation Ambient Lux Level: 188000 Lux Total Error: 15 % Signal Integration Time: 800 ms (maximum); 6.25 ms (minimum)
3-axis accelerometer sensor	ADXL346	Analog Devices	User selectable Measurement Range: $\pm 2g \pm 4g, \pm 8g, \pm 16g$ Output Resolution: 10-bit Sensitivity at $X_{OUT}, Y_{OUT}, Z_{OUT}$: $256 \text{ LSB}/g$ Scale Factor: $3.9 \text{ mg}/\text{LSB}$

1. THE INTERNET OF THINGS TECHNOLOGY

to support any extensive study and experimental evaluation of novel solutions for LLN [25].

The OpenWSN protocol suite is able to run on two native operating systems. The legacy kernel is *openos*, which is a single-threaded non-preemptive run-to-completion operating system. Its features are very far from a real-time operating systems, but its lightweight nature allows the stack to run also on very constrained devices (such as the TelosB described in 1.4.1.1. Starting from the last releases, also the *FreeRTOS* operating system has been integrated.

FreeRTOS is a full-featured preemptive task scheduler developed by Real Time Engineers Ltd and available at <http://www.freertos.org/>. The power and features of FreeRTOS kernel can be exploited for more advanced boards (such as the OpenMote-CC2538 described in Section 1.4.1.3).

Without loss of generality, it is possible to assume that the OpenWSN framework is divided in two parts: the firmware and the software, providing different and complementary functionalities.

The firmware is the portion of the stack that runs over the motes, and contains the protocol stack. It is written in the C programming language. In Figure 3.1 the OpenWSN default protocol stack is illustrated, explicating all the standards applied in.

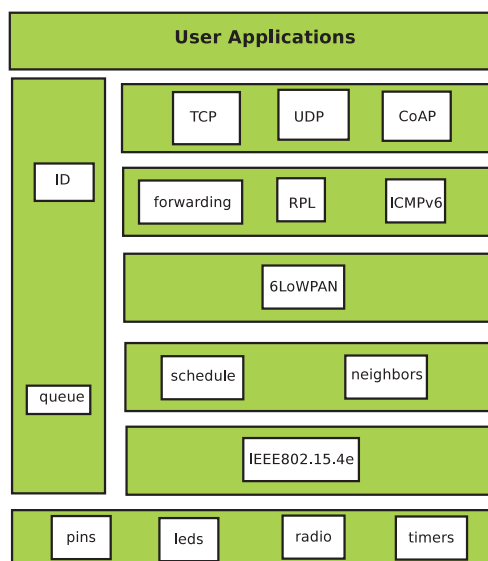


Figure 1.12: The OpenWSN default protocol stack.

The *Cross-layer* module contains some structures and methods, i.e. the queue and the `id_manager`, that can be exploited by all the procedures at different layers of the protocol stack. At PHY layer there are many modules, responsible for handling pins or leds of a particular hardware platform, or the communication with the radio module of the hardware used, and various timers used to regulate the functions as they have been performed. At MAC-low layer OpenWSN implements IEEE802.15.4e, which arises as a software improvement regards to the historical MAC protocol (i.e. the IEEE 802.15.4 standard described in section 1.3.2) without changing the underlying physical layer. Moreover, the IEEE 802.15.4 TSCH mode, 6LoWPAN and RPL protocols have been implemented as the reference protocols for the MAC, Adaption and Network Layer, respectively. At the transport layer, the OpenWSN protocol stack gives the choice to adopt TCP or UDP as the transport protocol. However, in a LLN context is preferred using the UDP protocol: in fact it provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism and overhead. Finally, the CoAP protocol designed in CoRE WG, is used the lightweight application layer.

It is worth noticing that, differently from the reference document [22], OpenWSN defines a default time-slot duration of 15 ms. This choice is due to the required compatibility with a wide range of platforms, some of them having slow processing and computational capabilities that make them unable to meet the 10 ms slot duration requirement defined in the IEEE 802.15.4 specification.

It is important to underline that the OpenWSN stack, in its original version, does not support any security feature, neither what included in IEEE 802.15.4 nor any application-layer security operation. These aspect will be investigated more deeply in the following chapters of this thesis.

As regarding the software, it includes the OpenVisualizer tool. It is written in Python programming language and provides three main functionalities: (i) it includes the upper portion of the protocol stack (from IP to CoAP) running on the PAN Coordinator, (ii) it offers gateway functionalities to the underlying IoT network, and (iii) it provides a tool for debugging and monitoring motes connected via USB to the host PC.

To provide a further insight, Tab. 1.2 reports the main hardware platforms running the OpenWSN protocol stack [25].

1. THE INTERNET OF THINGS TECHNOLOGY

Table 1.2: Platform running the OpenWSN protocol stack.

Hardware Platform	Manufacturer	Architecture	Maximum Speed	Flash	RAM	Radio Module
TelosB	Texas Instruments	16-bit	8 MHz	48 kByte	10 kByte	CC2420
GINA	Texas Instruments	16-bit	16 MHz	116 kByte	8 kByte	AT86RF231
WSN430	SensLab	16-bit	8 MHz	48 kByte	8 kByte	CC1101 or CC2420
Z1	Zolertia	16-bit	16 MHz	92 kByte	10 kByte	CC2420
OpenMote STM	Texas Instruments	32 bit	72 MHz	256 or 512 kByte	up to 64 kByte	AT86RF231
OpenMote CC2538	Texas Instruments	32-bit Cortex M3	32 MHz	up to 512 kByte	up to 32 kByte	CC2538
STM32 F103RE	ST Microelectronics	32-bit	72 MHz	512 kByte	64 kByte	AT86RF231
K20	FreeScale	32-bit	72 MHz	256 kByte	64 kByte	AT86RF231
MC1321x	FreeScale	8-bit	8 MHz	up to 60 kByte	up to 4 kByte	689S08A
EZ430-RF2500	Texas Instruments	16-bit	16 MHz	up to 32 kByte	2 kByte	CC2500

1.4.2.2 Contiki

Contiki¹ is an open-source and community-driven operating system for IoT development and prototyping.

While OpenWSN is a full protocol stack, a running Contiki system consists of the operating system core (with libraries and drivers) and the loaded programs. The core of the operating system is represented by a single binary, while each single loaded program can be distributed independently from the main operating system. The Contiki kernel offers no hardware abstraction. Therefore, libraries and/or drivers must provide this hardware abstraction by themselves. Moreover, all components of a Contiki application have direct access to the underlying hardware.

An important aspect to consider when using the Contiki system is related to its MAC protocol, i.e. the Contiki X-MAC.

It supports broadcast traffic, phase awareness, stream transport and optional MAC layer retransmissions. Moreover, it coordinates nodes in maintaining a strict schedule, during which they can switch their radio on. However, the MAC protocol is based on the concept of *strokes*. A stroke is, basically, a sequence of bytes, signalling the upcoming transmission by a neighbor device. The nodes listens for incoming strokes from neighbors, switching their radio off if no strokes are received and turning the radio off, after a successful communication, during the rest of the duty cycle period. If a stroke is received, instead, the node transmits a stroke acknowledgment packet to the sender. When the sender receives a stroke acknowledgment packet from the node to which it is sending stroke packets, the sender immediately sends it data packet.

This procedure is very different from the mechanisms described in IEEE 802.15.4 TSCH mode. Even if, in the last years, an extension of this mechanism have been included in Contiki to support the TSCH mode of the IEEE 802.15.4, this underlying mechanism is still present to manage the transmission/reception processes.

Similarly to OpenWSN, Contiki needs for 30 kByte of ROM and takes almost 10 kByte of RAM from the devices on which it runs. It is written in the C programming language and, differently from OpenWSN, it does not provide a software interface for handling live debugging functionalities. However, these functionalities can be still used during the developing phase of new version of protocols and algorithms, by means

¹<http://www.contiki-os.org>

1. THE INTERNET OF THINGS TECHNOLOGY

of instruments offered with the Cooja simulator. Finally, by considering the baseline time-slot duration as for OpenWSN (i.e., 15 ms), Contiki has a heavier radio interface management that brings to higher energy consumptions [43].

1.4.2.3 RIOT

In the last years the RIOT OS¹ emerges a new operating system, tailored to features and requirements of IoT devices.

The main rational behind its development is the requirement of porting features available on OS currently running on Internet hosts also on a target operating system for the IoT.

Its features includes energy-efficiency, small memory footprint, modularity, and uniform API access, independent of the underlying hardware [45].

RIOT OS is based on a microkernel architecture, that enable the support for multithreading using standard API. Moreover, it includes a TCP/IP network stack.

An important feature to be noticed is that there exist projects aimed at the integration of the OpenWSN protocol stack within the RIOT operating system. However, they are still in the design phase².

¹<https://www.riot-os.org/>

²<https://github.com/RIOT-OS/RIOT/tree/master/pkg/openwsn>

2

Securing IoT systems

In this chapter an overview of security issues and threats in the IoT context has been provided. Moreover, a deep analysis of the state of the art has been included, to highlight the evolution of security services in different IoT technologies, the main research directions of scientific communities and standardization activities, and to precisely identify which aspects need further investigations and analysis.

2.1 IoT security requirements

The main security services to be provided in a legacy Internet network still remain valid also in the IoT context. They include: Authentication of both communicating parties and messages, access control (i.e., authorization), confidentiality, messages integrity, non-repudiation and availability of services [46].

Authentication of communicating parties refers to the possibility for each device to demonstrate its identity in an incontestable fashion. This is usually achieved by forcing the device to use cryptographic keying material uniquely associated with its identity. In the same fashion, authentication of messages refers to the unique association of a given message to the entity which generated it. As for the previous point, a common approach to this issue is to associate to the message a digest created through cryptographic material in possession of the entity.

Authorization or, equivalently, access control, involves the verification that each entity in the system is authorized to perform a given action on a specific object. An access control mechanism should prevent the unauthorized use of resources, by strictly

2. SECURING IOT SYSTEMS

controlling who can access resources, what a given entity can do and in what conditions. It often leverages the same kind of mechanisms used for authentication, while requiring strict interacting with the system administrator.

The data confidentiality refers to the possibility to limit the access to information stored within packets to only authorized nodes. It is generally achieved by encrypting part of the message through the adoption of key and algorithm common to both sender and receiver. While not knowing secrets exploited by the encryption process, a given attacker could modify the content of the message while it transits over the network for reaching the destination node.

The message integrity protection must ensure that the packet arrives at the receiver side without any tampers and alterations. This is usually achieved thanks to the use of a one-way hash function that, by combining all the bytes in the message with a secret key, is able to verify the integrity of the packet itself [25].

Non-repudiation service, instead, provides protection against potential misbehavior by one of the communicating parties. Focusing on the sender of a given message, this service provide the proof that a message has been delivered by a given entity. Looking at the destination side, it provides a proof that a message has been received by a given party.

Finally, the availability of a service is the property of a system or a resource, to be always accessible and usable upon demand by an authorized system entity, according to performance specifications for the system.

2.2 Practical limitations in real IoT systems

Securing IoT systems could be a really challenging task, given the plethora of constraints and requirements that should be considered at the same time when designing a suitable solution.

The devices that create the IoT are usually constrained, both in terms of processing capabilities and in memory/storing availability. Therefore, from one side running heavy and time-consuming protocols is not a feasible option while, from the other side, storing a wide range of secrets and security-related data could not be an efficient solution.

Moreover, the vast majority of protocols and algorithms currently running on the Internet could not be a suitable choice, because of the number of messages that they

2.3 Challenges for a secure IoT

involve. In fact, given that IoT devices are usually battery-powered, they should reduce at maximum the number of radio operations and, consequently, the messages exchanged in the network. Many protocols, such as IPSec or TLS, create a secure link through heavy transactions, very hard to be managed by a constrained device.

Finally, it is worth noting that a number of limitations exist not only from a practical perspective (i.e., looking at the devices' capabilities), but also from a protocol perspective. In fact, protocols such as TLS, IKE, IPSec or EAP involve long messages, that can be forced to fit into the limited MTU available for IoT devices only through heavy fragmentation. This clearly increases the packets exchanged in the network, often through lossy, low-power and short-range wireless links. This aspect further increase the demand for lightweight and energy efficient solutions.

2.3 Challenges for a secure IoT

A wide set of security risks exist when dealing with IoT networks. As discussed in [47] and summarized in Table 2.1, they can be related to a particular phase in the device life-cycle.

Table 2.1: Threats in the life-cycle of an IoT device.

Layer	Manufacturing	Installation	Operation
Transport/ Application Layer	-	Eavesdropping	Eavesdropping Man In The Middle Unauthorized access
Network Layer	-	Man In The Middle	DoS, Routing attacks
Physical Layer	Device Cloning	Substitution Steal	Side-channel, Tamper, Fault Extraction of security parameters

Looking at the physical layer, the devices that creates the IoT are usually tailored to specific tasks, and, often, they are created by different manufacturers. Attacks that

2. SECURING IOT SYSTEMS

occurs in the manufacturing phase usually an untrusted manufacturer that clones the device. The cloned device can be simply sold for a cheaper price but, in a worst case scenario, the firmware it includes could be changed to implement harmful features (i.e., it can contain backdoors or other malicious software).

During the installation phase, devices are usually provided with a unique identity and cryptographic materials, used for subsequent communications. An attacker could substitute the device, or it can steal the device's secrets, thus compromising its functionalities.

Focusing on the operation phase, it is possible to consider passive and active attacks. Passive attacks include the gathering of information related to the particular device under attack, without any on-line physical connection to the device by the attacker. When performing active attacks, instead, the attacker interacts with the device in an explicit fashion. Three categories of attacks can be considered in the IoT context, namely tampering and fault attacks for what concerns active attacks, and side channel passive attacks.

With the tamper attack, a malicious user tries to physically connect to the device for extracting the key. To protect against this attack, specific software and/or hardware based techniques can be used to prevent the physical access to security parameters stored within the device.

With the fault attack, with reference especially to public key algorithms, a malicious node forces the victim device to perform calculations on weak groups elliptic curves in order to leak the secret key. To avoid this issue, each device should accept a restricted number of trusted mathematical groups or elliptic curves, known to be resilient against fault attacks.

With a side-channel attack, finally, it is possible to determine the secret key starting from side-channel information like power consumption, electromagnetic radiations, running times, error messages associated to the hardware computation.

Focusing on the network layer, a wide set of threats emerge both in the installation and in the operation phase.

Specifically, eavesdropping on the wireless communication channel involves a passive listening of all the communications between a given couple of devices, and it is particularly hard to notice because of the total absence of interactions originated by

the attacker. In this sense, encrypting all the communications with a dedicated key and an algorithm could be a useful countermeasure.

Denial of Service (DoS) attacks are typically performed by powerful nodes that creates a lot of requests for target devices, thus saturating its bandwidth and its reachability by other devices and excluding it from the network.

Man-In-The-Middle (MITM) is a well-known attack, that arise from the lack of mutual authentication in the communication. The attacker can relay and alter the communication between two nodes, without any possibility to be detected. As an attack that exploits the lack of mutual authentication, or some weaknesses carried in this process, a MITM attack can be successful only if the attacker is able to impersonate each endpoint to its satisfaction, as expected from the legitimate other party.

Routing attacks includes a wide set of threats. To name a few, *Node Impersonation* occurs when an attacker gains access to a network behaving and declaring itself as a legitimate node. Moreover, *confidentiality attacks* at the network layer aim at exposing routing information or routing exchange data. *Exposure attacks* involve a routing entity to expose information to an outside component without any cryptographic protection. In *passive wiretapping* attacks the malicious node listens communications and information exchanged between nodes. By analyzing traffic, the attacker can learn about the network topology. Attackers can also map the network by analyzing the traffic flow throughout a network. By analyzing simply source and destination addresses, attackers can map the network and flow patterns, and originate other active attacks such as *overclaiming and misclaiming attacks*, which aims at changing the topology by injecting false routing messages in the network. Another class of routing attacks is *Availability Attacks*, in which the attacker can both compromise a group of nodes by means of DoS or Distributed DoS (DDoS), or injecting false routing information in the network, thus forcing the data to traverse compromised nodes to reach the intended destination. *Sinkhole attacks* leverage a compromised node (typically an inside node in a given network topology) to advertise good links and attract traffic.

Finally, it is worth citing also *Byzantine routing attacks*, in which a compromised node still continues operating legally in the network, by using its identity and keying materials, but being completely controlled by the attacker.

The issues previously identified and connected to the network layer are still present also at the application layer. Therefore, effective countermeasures able to protect the

2. SECURING IOT SYSTEMS

communication against eavesdropping, MITM and DoS attacks must be put in place also at the application layer. Moreover, unauthorized access to resources represents another issue to deal with, especially at the application layer, given the fact that the most of requests come directly from the Internet. [48].

2.4 Security Algorithms for IoT

In the following a brief overview of the main symmetric and asymmetric cryptography techniques available and used in embedded devices is carried out. The following algorithms are at the basis of the approaches leveraged in literature and in following sections of this thesis.

2.4.1 Symmetric Cryptography

Generally speaking, symmetric cryptography techniques involve the use of the same key for the encryption and the decryption process. This key must be known only to the the sender and the destination of the message, and its disclosure compromises the security of the whole cryptosystem.

Five different ingredients can be found in a symmetric encryption scheme [46]:

- Plaintext: the original message that needs to be protected.
- Encryption Algorithm: it is an algorithm that performs various substitution and permutation on the plaintext;
- Secret Key: it is a value (usually a string), independent from the algorithm, that is known only to the sender and the destination. Using different secret keys with the same encryption algorithm on the same plaintext provides a different output.
- Ciphertext: it is the output of the encryption algorithm, and depends on the plaintext, the encryption algorithm and the secret key. It appears as uncorrelated with the plaintext.
- Decryption Algorithm: usually, it is the encryption algorithm running in inverse mode. By applying the decryption algorithm on the ciphertext using the same secret key used for the encryption algorithm, it is possible to retrieve the original plaintext.

Among the wide set of symmetric algorithms available in literature, in the following only the AES algorithm is described. This is not only because it is nowadays considered the most secure symmetric technique, but also because the vast majority of IoT devices offer the possibility to perform it in hardware, by using dedicated registers or buffers. As will be clearly showed by experimentation in Section 3, this feature provides significant advantages in terms of execution time and energy consumption.

2.4.1.1 The AES algorithm

The AES algorithm is a block cipher, standardized in 2001, which encrypts a sequence of plaintext bytes in blocks of 16 bytes. The key used to encrypt the block can be 128, 192 or 256 bytes long.

It uses a series of permutations and substitutions, and therefore executes fast when implemented in both hardware and software.

AES involves the execution of 10 rounds. Each round consists of several steps, made up of four stages, including one that depends on the encryption key itself. As for the decryption process, some reverse rounds are applied to recover the plaintext from the ciphertext, by applying the same encryption key.

Each 16-bytes input vector in AES is represented through a 4×4 square matrix, in the form:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix}$$

The four stages are:

- *SubBytes*: by using an 8-bit substitution box, namely the Rijndael S-box, each byte of the state matrix $a_{i,j}$ is replaced with a substituted byte SubByte, $S(a_{i,j})$.
- *ShiftRows*: each element of a given row in the state matrix is shifted on the left by an offset, that is equal to the row index minus 1. In general, if i is the number of the row in the matrix, ($offset_i = i - 1$).
- *MixColumns*: it uses an invertible linear transformation to combine the bytes in each column between them.

2. SECURING IOT SYSTEMS

- *AddRoundKey*: As the final stage in the step, the state matrix is combined (with an x-or operation) with the key for this step.

From an high-level perspective, instead, it is possible to identify four main processes:

- KeyExpansions: from the main key, it creates a series of 12 keys by using a *key schedule* algorithm. This set of keys are then used in each of the following rounds.
- Initial Round: it consists of a single *AddRoundKey* operation;
- 10 Rounds: in each of the rounds, the four operations previously described are executed (SubBytes, ShiftRows, MixColumns, AddRoundKey);
- Final Round: it is similar to the previous 10 rounds, but it does not include the MixColumns stage.

The decryption algorithm uses the inverse mode of each of the four stages previously described.

Finally, it is worth noting that AES can be executed also on 8-bit micro-controllers in a very efficient fashion, given that it operates through *x-or*, shift and mix operations, that can be implemented very simply also in hardware.

2.4.2 Asymmetric Cryptography

On the contrary with respect to symmetric cryptography techniques, the algorithms classified in the *Asymmetric Cryptography* group or, equivalently, *Public-Key Cryptography*, generally involve the use of two different keys, namely the private key and the public key, to be used for the encryption and the decryption process in an alternative way.

Its introduction is due to Diffie and Hellman in 1976 [49]. Public-key cryptography provided a significant breakthrough in cryptography, essentially for two reasons. From one side, asymmetric algorithms leverage mathematical functions rather than substitution and permutation. From the other side, they involve the use of two separate keys, in contrast to symmetric encryption, which uses only one key.

The five building blocks identified for the symmetric cryptography still exist in the asymmetric mode, with two significant differences:

- the key is not single anymore. It is decoupled in a private key, that must be known only to a given party, and a public key, that must be publicly available. In general, if one of these is used for encryption, the other is used for the decryption (note that both of them can be used for encrypting and decrypting a plaintext, providing different security services).
- the encryption algorithm and the decryption algorithm are based on mathematical functions rather than simple substitutions and permutations, and they apply various transformations to the plaintext.

More than representing an advance with respect to the symmetric cryptography techniques, asymmetric cryptography is a concrete alternative. In fact, public key cryptography algorithms can be used for three different purposes:

- Confidentiality: in this case, the sender encrypts the message by using the destination's public key. The receiver, decrypts the message by using its private key.
- Authentication (Digital Sign): in this case, the sender *signs* the message through its private key. Any receiver of the message can verify its authenticity by using the public key of the source.
- Key distribution: in this case, the two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties [46].

Despite these interesting functionalities, the operations involved by Asymmetric algorithms are heavier and more time and energy consuming than symmetric mechanisms.

For these reasons, for years they have not been considered a feasible option for constrained devices. Nevertheless, the advances in embedded technologies carried out in the last years conducted to a new examination of these techniques, especially in solving the key distribution problem [50].

2. SECURING IOT SYSTEMS

2.4.2.1 RSA

The Rivest-Shamir-Adleman (RSA) scheme is one of the most widely accepted and implemented general-purpose approach to public-key encryption. Basically, it is a block cipher in which, given n as the size of the reference group, the plaintext and ciphertext are integers between 0 and $n - 1$. A typical size for n is 1024 bits or, equivalently, 309 decimal digits.

RSA leverages the observation that there exist three very large positive integers e, d and n such that:

$$(m^e)^d \equiv m \pmod{n}. \quad (2.1)$$

Moreover, even knowing e and n or even m it can be extremely difficult to find d .

Suppose two entities A and B wants to securely exchange a message m , according to the RSA scheme. It involves four steps: key generation, key distribution, encryption and decryption [46].

In the key generation phase, the public and private coefficients e and d are generated. In detail, starting from two random prime numbers p and q , firstly n is computed as $p \cdot q$, then $\varphi(n)$ is calculated as $(p - 1) \cdot (q - 1)$. Starting from these values, the public key e is chosen such as the greatest common divisor (gcd) between e and $\varphi(n)$ is 1, i.e. they are co-prime. The private key d is the inverse of e modulo n , i.e. $d = e^{-1} \pmod{n}$.

Next, to enable B to send its encrypted message, A delivers its public key, that is the couple (e, n) . Given these values, B can encrypt the message m by using Equation 2.2:

$$c = (m^e) \pmod{n}. \quad (2.2)$$

Then, B delivers c to A. At this point, A can decrypt message by using Equation 2.3.

$$m = (c^d) \pmod{n} = (m^e)^d \pmod{n} = m \pmod{n}. \quad (2.3)$$

It is worth noting that the security of the scheme leverages the difficulty for an attacker to calculate the private key d , even knowing the public key e and the modulus n . This is true especially for very large values of the couple e, n . This is the reason

why asymmetric cryptography techniques such as RSA are as more secure as keys are longer.

Finally, note that RSA is often used for exchanging keys between two entities (i.e., M is a symmetric key that is then used to secure communications using a symmetric encryption scheme, as AES) or to verify the authenticity of communicating parties.

2.4.2.2 Diffie-Hellman Key Exchange

The Diffie-Hellman (DH) key exchange is the first historical public key algorithm, developed in 1976 [49]. Its purpose is to enable two users to securely exchange a secret, that can then be further used to generate a shared key for a symmetric encryption scheme. In its basic version, the algorithm is limited to the exchange of secret values. Therefore, its effectiveness relies on the difficulty for the adversary of computing discrete logarithm, i.e. to find a value i such that, given three integers α , b and q , the Equation 2.4 is valid [46].

$$b = \alpha^i \pmod{q}. \quad (2.4)$$

More in detail, the value α must be a primitive root, i.e. the values $\alpha \pmod{q}$, $\alpha^2 \pmod{q}$, ..., $\alpha^{q-1} \pmod{q}$ are distinct and equal to values from 0 to q , in some permuted order.

Generally speaking, the DH algorithm involves two steps: the keys generation and the secret computation.

In the setup phase, the involved entities agree on a prime number q and a primitive root α . Starting from these shared values, both the entities A and B can choose their private keys, X_A and X_B , respectively, such that $X_A < q$ and $X_B < q$.

Next, both can generate their public keys as in the Equation 2.5.

$$\begin{aligned} Y_A &= \alpha^{X_A} \pmod{q}, \\ Y_B &= \alpha^{X_B} \pmod{q}. \end{aligned} \quad (2.5)$$

Each node keeps the X value private and makes the Y value publicly available. Both of the users can then calculate a shared secret, K_A and K_B , by using the Equation 2.6.

2. SECURING IOT SYSTEMS

$$\begin{aligned} K_A &= Y_B^{X_A} \pmod{q}, \\ K_B &= Y_A^{X_B} \pmod{q}. \end{aligned} \tag{2.6}$$

It is trivial to verify that the two calculations reported in Equation 2.6 produce identical results [46].

The security of the key exchange lies on the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For very large primes, this task is widely considered unfeasible.

Finally, note that in this very basic version the exchanged public coefficients are not authenticated. Therefore, the MITM attack would be feasible and successful. That's why the DH approach is often complemented with at least another message exchange needed to verify the authenticity of the involved parties.

2.4.3 Elliptic Curve Cryptography

The most of public key cryptosystems are based on the RSA algorithm. Even if its security has been proved and widely accepted, the key length to be adopted increased over recent years, while creating a heavier processing load on applications using it. Recently, a competing system has begun to challenge RSA: Elliptic Curve Cryptography (ECC).

Comparing ECC to RSA, it offers equal security level for smaller key sizes, thereby reducing processing overhead. On the other hand, ECC products started to appear only in last years, and therefore the interest in probing its weaknesses has just begun. Accordingly, the confidence level in ECC is not yet as high as that in RSA.

From a mathematical perspective, elliptic curves are not ellipses. Their name derives from their description through cubic equations (similar to those used for calculating the circumference of an ellipse), which takes the form in Equation 2.7.

$$y^2 + axy + by = x^3 + cx^2 + dx + e, \tag{2.7}$$

where a, b, c, d , and e are real numbers and x and y are values in the real numbers. In the following the discussion will be limited to equations in the form defined by Equation 2.8.

$$y^2 = x^3 + ax + b. \tag{2.8}$$

In the definition of an elliptic curve a single element denoted O is always included, namely the point at infinity or the zero point.

ECC leverages elliptic curves in which the variables and coefficients are restricted to elements of a finite field [46]. Specifically, in cryptographic applications it is possible to find *prime curves*, defined over Z_p with p that is a prime number, and *binary curves*, defined over $GF(2^m)$.

Prime curves over Z_p uses a cubic equation in which the variables and coefficients assume values in the set of integers from 0 to $p - 1$, and in which calculations are performed modulo p . Binary curves defined over $GF(2^m)$, instead, includes variables and coefficients that are values in $GF(2^m)$ and in calculations are performed over $GF(2^m)$.

It is possible to demonstrate that the addition operation in ECC (namely, elliptic curve point addition) is equivalent to a modular multiplication in RSA, while multiple addition operations or, equivalently, elliptic curve point multiplication, is the counterpart of modular exponentiation.

As in RSA the *hard problem* for an attacker is related to find the discrete logarithm of a value, the equivalent hard problem in elliptic curves is called *discrete logarithm problem for elliptic curves*. In fact, by considering the equation $Q = kP$ in $GF(2^m)$, where $Q, P \in E_p(a, b)$ and $k < p$, it is relatively easy to calculate Q given k and P , but it is computationally hard to determine k given Q and P , especially for great values.

2.4.3.1 Elliptic Curve Diffie Hellman

Elliptic Curve DH (ECDH) is an anonymous key agreement protocol, that allows two parties, each of them having a private-public elliptic curve key pair, to establish a shared secret over an insecure channel [51].

In the setup phase, the two involved parties agree on an elliptic curve $E_q(a, b)$, with a and b that are elliptic curve parameters, and q that is a prime number in the form 2^m . Moreover, they agree on a base point $G = (x_1, y_1) \in E_p(a, b)$, with order n , chosen such that $nG = O$.

As for the legacy DH approach, it is possible to distinguish the key generation and shared key computation phases.

In the Key Generation phase, both of the peers chooses two secret private numbers, n_A and n_B , respectively, chosen such that $n_A, n_B < n$. Starting from these values,

2. SECURING IOT SYSTEMS

they can generate their respective public keys as in Equation 2.9, where the operator $\times$ refers to the point multiplication.

$$\begin{aligned} P_A &= n_A \times G, \\ P_B &= n_B \times G. \end{aligned} \tag{2.9}$$

The two peers exchange their public values. Both of the users can then calculate a shared secret, K_A and K_B , by using the Equation 2.10.

$$\begin{aligned} K_A &= n_A \times P_B, \\ K_B &= n_B \times P_A. \end{aligned} \tag{2.10}$$

Also in this case it is possible to demonstrate that the calculation started by the two peers lead exactly to the same solution, as shown in Equation 2.11.

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A. \tag{2.11}$$

To break this scheme, an attacker would need to be able to compute K_A (or K_B) given G and P_B (or P_A), which is computationally hard.

It is worth noting that ECDH provides the same security level offered by the legacy DH scheme by using smaller keys. However, it still suffers from the same threats already discussed in Section 2.4.2.2.

2.5 The evolution of security systems for IoT

Since their first appearance on the research and market side, security issues have appeared as a blocking concern for IoT technologies. In what follows an overview is provided of the main security mechanisms designed and put in place for a range of technologies, starting from WirelessHART to Zigbee IP solutions.

2.5.1 Security in WirelessHART and ISA 100.11a

For historical reasons, WirelessHART and ISA 100.11a technologies have been the first industrial standards to face the communication security problem. WirelessHART and ISA100.11a applied confidentiality protection through payload encryption and message authentication for both single-hop, hop-by-hop and end-to-end messages [19]. Security countermeasures are applied both at the MAC layer, against attackers outside the

2.5 The evolution of security systems for IoT

system, and on the transport and application layer with respect to attackers on the path between the source and the destination of a given message. The AES-CBC-MAC Mode (CCM) scheme is applied both in WirelessHART and ISA 100.11a technologies, to provide authentication and privacy, and both of them relies on a set of pre-shared keys directly loaded within the devices. This key, so called the *join key*, is initially used by a joining device to obtain a new key, namely the *session key*, used for further communications.

In WirelessHART devices are equipped with a network key and a session key, in addition to the join key previously described. The session key is used for authentication purposes at the network layer, and dedicated session keys can be provided for each communication link. The network keys, instead, are used at the MAC layer to authenticate hop-by-hop messages and they can be rotated based on the security procedures of the process automation plant.

In ISA 100.11a installations the security architecture is more complex. The use of the join key is optional, while two other keys, namely the Master Key and the DL Key, are mandatory and another key, the session key, is optional. While the master key is used to secure the link between the security manager and the device, the DL key is used at the MAC layer for the computation of the Message Integrity Code (MIC) associated to the message; finally, if the device has been provisioned with a dedicated session key, the MAC payload is encrypted by using this key. As in WirelessHART, the keys are limited in time and need to be periodically updated. In addition, ISA100.11a also supports optional asymmetric keys, that can be used to protect messages. If asymmetric cryptography techniques are enabled, a certification authority must be deployed in the system to assist in verifying the true identity of the device communicating the certificate.

Depending on the chosen technique, an ISA 100.11a installation can run in three different security modes: (i) no security, (ii) symmetric keys, and (iii) public keys. It is worth noting that also the *no security* mode involves the usage of a key, that is a widely known global key, while the MIC over MAC messages is calculated through a simple Cyclic Redundancy Check (CRC), without any security guarantee.

2. SECURING IOT SYSTEMS

2.5.2 Bluetooth security

From a security perspective, a Bluetooth device can operate in one of the following three security modes: (i) no security, (ii) service-level security (in which authentication and encryption can be applied when a given service requires them) or (iii) device-level security, in which each communication must be secured starting from the connection instant [27].

The *pairing* is the procedure that allows two Bluetooth devices to establish a secure connection at their first join to the network.

During the pairing procedure devices generate and exchange their link keys and stores them in hardware to reuse them in further connections. When devices are paired they are said to be *Paired* or *Bonded*.

The specific pairing procedure mainly depends on the specific radio included in a given device, i.e. the specification it implements.

Bluetooth devices related to the Core Specifications (prior version 2.1) execute the *Legacy Pairing*, designed for very constrained devices. They use the SAFER+ algorithm for the key derivation, while the E0 stream cipher is used for the encryption of the message. Authentication keys are 128 bits long and they are generated starting from PIN codes, usually only 4-bytes long. Moreover, the E0 encryption cipher has been widely recognized as a weak mechanism and the link keys their selves are not rotated nor re-generated, being susceptible to very trivial brute force attacks. Finally, note that this pairing mechanism does not include any peer authentication, thus being susceptible to MITM attacks.

The *Secure Simple Pairing (SSP)* has been introduced from version 2.1 and it is mandatory to implement starting from devices in this family. SSP leverages the ECDH scheme, and further allows to protect against MITM. It can be run by any Bluetooth device, independently from its capabilities, even if some differences (and offered security functionalities) exists with respect to Input/Output (I/O) capabilities of the involved devices. Specifically, the following models can be chosen according to the available I/O functionalities [27]:

- *Numerical Comparison*: if the devices have sufficient capabilities to display a six digits number and type a YES/NO input, they can compare these on the screen and type YES or NO if they matches or not, respectively. Note that even knowing

the group of six digits does not provide any possibility to decrypt the ciphertext between the two devices.

- *Just Works*: targeted for very constrained devices, which do not have any I/O capability. The functioning is the same with respect to the previous model, but it does not require any user intervention. Therefore, it is aimed to work on devices that does not have the possibility to interact with the user in real-time (i.e., they do not have a screen and a keyboard to perform the previous steps).
- *Passkey Entry*: designed for specific scenarios in which one side needs only a screen, while the other side has at least a keyboard. A six digit number is shown on the screen side and the other part must input this number on the keyboard. Note that, differently from the Legacy Pairing previously described, here the six digits are just an artifact of the algorithm, but they does not provide randomness (as was true, instead, for the legacy pairing)
- *Out of Band*: if devices have the possibility to use another technology, such as Near Field Communication (NFC), they leverage the capabilities there available to secure also the Bluetooth link.

While Bluetooth High-Speed (v3) still leverages the SSP scheme, Bluetooth LE (v4.0) gets rid of the ECDH scheme for reducing energy consumption, and implements the AES-CCM mechanism, performed in the controller, with keys generated in the host (i.e., in software), with customized mechanisms. No protection against passive eavesdropper is provided.

2.5.3 Security in Zigbee IP

With reference to security issues, Zigbee IP defines key distribution and joining procedures, that are managed, in a centralized fashion, by the *Trust Center* [52]. In particular, it introduces three different keys: the Master key, used to finalize authentication procedures; the Network key, for protecting management messages; and the Link key, adopted to protect the communication between two devices. Generally, both Master and Network keys can be pre-configured or delivered by the *Trust Center*. The Link Key can be either distributed by the *Trust Center* or negotiated between two devices under the supervision of the *Trust Center*. In details, the key agreement is

2. SECURING IOT SYSTEMS

handled by means of the Symmetric-Key Key Establishment (SKKE) protocol [53]. It uses a centralized scheme where the *Trust Center* always coordinates the negotiation of keys among any couple of nodes.

As reported by the message sequence chart in Figure 2.1, the child node sends a first request to the *Trust Center*, which replies both to child and parent nodes with an initial shared secret (i.e., the *Master Key* in the SKKE protocol language). Then, this secret is used to protect the following 6 messages exchanged between the nodes, for finalizing the key agreement mechanism. Now, considering the size of each logical message and the constraint on the MTU, 9 different MAC messages are required to complete the protocol.

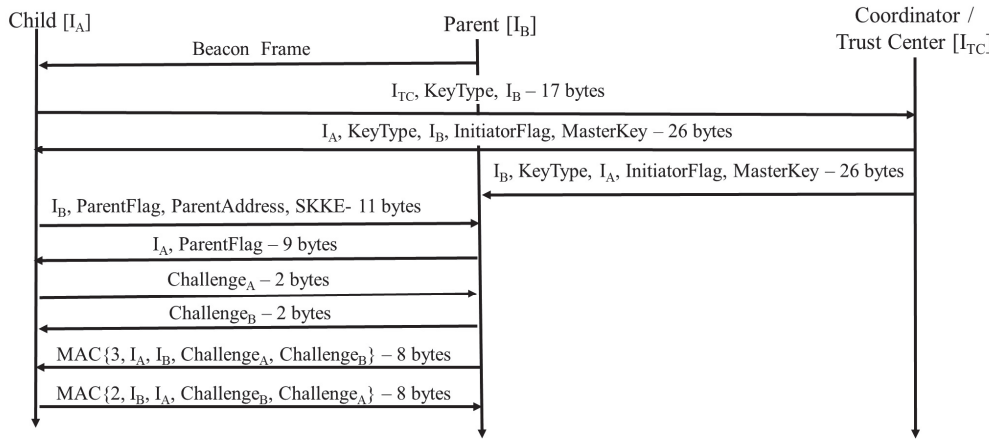


Figure 2.1: Establishing a secured communication in ZigBee IP specifications.

It is possible to observe that the SKKE protocol integrated in ZigBee IP specifications does not support an authenticated key agreement mechanism. In addition, in Zigbee IP some messages (i.e., the second, the third, and fourth in Figure 2.1) are exchanged with the *Trust Center* that may be some hops away from the nodes involved in the KMP. This negatively impacts on energy consumptions and latencies.

Finally, Figure 2.1 shows that the SKKE protocol requires a total number 9 messages to establish a shared key. It may be not really scalable, especially in large IoT systems, and it may suffer fault-tolerance issues because the *Trust Center* is a single point of failure [54].

2.6 State of the art on security protocols for IoT networks

In what follows a big picture of the current research activities around IoT security is provided, focusing on both standardization activities and scientific literature. The analysis will be conducted following a bottom-up approach in the IETF 6tisch protocol stack described in Section 1.3.5.

2.6.1 MAC layer security

By observing IoT systems from the bottom, securing communications at layer-2 represents the first important countermeasure to security threats [50]. In this context, the IEEE 802.15.4 standard emerges as the leading enabling technology for short-range low-rate wireless communications. Herein, the most significant contributions related to MAC-layer security will be summarized, with a major emphasis on IEEE 802.15.4-based solutions.

2.6.1.1 Security in IEEE 802.15.4

The IEEE 802.15.4 specification introduces procedures and mechanisms for protecting MAC frames, through symmetric-key cryptography techniques based on the *AES-CCM** algorithm. In the case security features are supported by a given device, the *macSecurityEnabled* attribute, stored at the MAC layer, is set to TRUE.

Eight security levels are defined to protect the frame generated at the MAC layer in different manners. As summarized in Tab. 2.2, they include *unsecured*, *only encrypted*, *only authenticated*, and *encryption with authentication* configurations. When the *unsecured* level is enabled, nor data confidentiality neither message integrity are provided. In other cases, instead, the data encryption and the authentication of messages are provided by means of AES and AES-CBC techniques, respectively. It is possible to offer a specific service to each kind of packet. However, the selection of the security level and the definition of other parameters required for performing security procedures have to be handled by an upper layer and then communicated to the MAC entity through dedicated primitives.

The IEEE 802.15.4 MAC frame, which has been pictured in Fig 2.2, is composed by a MAC header, a payload and a Frame Check Sequence (FCS) footer. Security

2. SECURING IOT SYSTEMS

Table 2.2: Different security levels provided by the IEEE802.15.4 standard.

Security Level	Security Level Field b2, b1, b0	Security Attributes	Data Confidentiality	Data Authenticity	Authentication tag length (bytes)
0	000	none	OFF	NO	0
1	001	MIC-32	OFF	YES	4
2	010	MIC-64	OFF	YES	8
3	011	MIC-128	OFF	YES	16
4	100	ENC	ON	NO	0
5	101	ENC-MIC-32	ON	YES	4
6	110	ENC-MIC-64	ON	YES	8
7	111	ENC-MIC-128	ON	YES	16

parameters are included within the *Frame Control* and the *Auxiliary Security Control* fields.

If the *Security Enabled* flag of the *Frame Control* field is set to 1, it means that the current MAC frame is protected and the node transmitting the packet supports at least one of the security services discussed above. If the flag is set to 0, instead, it means that the device sending the packet does not support any security capabilities and, for this reason, it is not able to send and receive encrypted and/or authenticated messages.

The *Auxiliary Security Control* field, which is present into the MAC header only if the *Security Enabled* flag is equal to 1, stores some parameters adopted for protecting the frame. They will be exploited by the destination node for performing the reverse security procedure (i.e, decryption and/or integrity check). This security header is composed by the *Security Control*, the *Frame Counter*, and the *Key Identifier* fields. The first one explains the security level and the key identification mode chosen by the sender. The counter stored into the second field is generated by the source in order to protect the message from replay attacks. Finally, the last field, i.e., the *Key Identifier*,

2.6 State of the art on security protocols for IoT networks

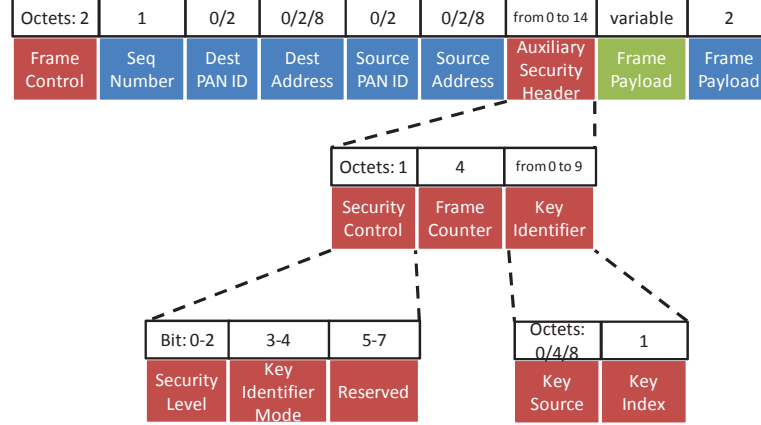


Figure 2.2: The Auxiliary Security Header Structure.

is optional and it stores information (*KeySource* and *KeyIndex*) needed to determine the key exploited for the encryption of the message.

At the MAC layer, encryption and decryption functionalities are implemented within the *outgoing frame security* and the *incoming frame security* procedures, respectively. They exploit a number of security attributes, which have been summarized in Tab 2.3.

It is very important to remark that the standard allows the possibility to use a dedicated key for each remote device and for each type of MAC frame (i.e., beacon, command frame, data packet, and ACK). Moreover, it is necessary to define a specific security service to be guaranteed for each kind of message. The related information is stored in the *macSecurityLevelTable*. It is composed by a set of *SecurityLevelDescriptor* elements, which provide information about the frame type which it refers to, the minimal expected/required *security level*, the set of allowed *security levels*, and a boolean flag indicating if the minimal security service may be overridden by a given device.

A node stores into the *macDeviceTable* the list of devices with which it can setup a secure communication. For each of them, a dedicated *DeviceDescriptor* is created. It contains the PAN ID, its short MAC address, its extended MAC addresses, as well as the counter of the latest packet received from the remote device and a boolean flag indicating if the considered node may override the minimum security level settings.

Without any doubts, the most important attribute is the *macKeyTable* where all the keys are organized in. A *keyDescriptor*, i.e., the single element of the aforementioned

2. SECURING IOT SYSTEMS

Table 2.3: Security-related attributes defined in IEEE 802.15.4.

Attribute	Type	Description
<i>macKeyTable</i>	Set of <i>KeyDescriptors</i>	It stores keys and other specific information, useful for protecting a MAC frame. A <i>keyDescriptor</i> element is created for each key, and contains: the key, the set of devices that can use it, a list of <i>KeyUsageDescriptor</i> indicating which frame may be protected with this key, and <i>KeySource</i> and <i>keyIndex</i>) parameters adopted to uniquely identify the key.
<i>macDeviceTable</i>	Set of <i>DeviceDescriptors</i>	It provides some information about remote devices which the node can establish a secure communication with. A dedicated <i>DeviceDescriptor</i> is associated to each remote device.
<i>macSecurityLevelTable</i>	Set of <i>SecurityLevelDescriptors</i>	It provides information about the security level required for each MAC frame type and subtype. In particular, each <i>SecurityLevelDescriptors</i> stores information about the frame type which it refers to, the minimal expected/required <i>security level</i> , the set of allowed <i>security levels</i> , and a boolean flag indicating if the minimal security service may be overridden by a given device.
<i>macFrameCounter</i>	Integer	The outgoing frame counter for the considered device.
<i>macAutoRequestSecurityLevel</i>	Integer	The security level used for automatic data requests.
<i>macAutoRequestKeyIdMode</i>	Integer	The key identifier mode used for automatic data requests. It is invalid if the <i>macAutoRequestSecurityLevel</i> attribute is set to 0x00.
<i>macAutoRequestKeySource</i>	Short or extended IEEE 802.15.4 MAC address	The originator of the key used for automatic data requests. This attribute is invalid if the <i>macAutoRequestKeyIdMode</i> element is invalid or set to 0x00.
<i>macAutoRequestKeyIndex</i>	Integer	The index of the key used for automatic data requests. It is invalid if the <i>macAutoRequestKeyIdMode</i> attribute is invalid or set to 0x00.
<i>macDefaultKeySource</i>	Extended IEEE 802.15.4 MAC address	The originator of the default key used for key identifier mode 0x01.

table, contains the key, the set of devices that can use it, a list of *KeyUsageDescriptor* indicating which frame may be protected with this key, and other parameters (e.g., *KeySource*, *keyIndex*) used for uniquely identifying the key.

A node that intends to secure a packet (i.e., by encrypting and/or authenticating the MAC payload) has to execute the *outgoing frame security* procedure, i.e., the following steps:

1. identify the *Security Level* that has to be applied to the current MAC frame among those listed in Tab 2.2;
2. make sure that the size of the frame does not exceed the maximum allowed value (i.e., 127 bytes [22]);
3. identify the key to use during the encryption process. It will be selected among those available into the *macKeyTable* by taking into account the *Key Identify Mode* announced by the upper layer;
4. protect the MAC payload according to the selected *Security Level*, by using the CCM* algorithm;
5. create the *Auxiliary Security Control* field and include it within the protected frame;
6. generate the FCS;
7. reassemble the whole packet;

When a device receives a MAC frame, it should firstly verify if it has been protected by the sender (i.e., if the *Security Enabled* flag is set to 1). In affirmative case, it will run the *incoming frame security* procedure, i.e., the following operations:

1. verify the packet integrity through the check of the FCS;
2. identify the key to exploit during decryption process;
3. verify that the *Security Level* chosen by the sender is allowed for the message the packet contains;
4. decrypt the payload;

2. SECURING IOT SYSTEMS

5. verify that all security constraints (e.g., allowed *Security Level*, allowed key, frame counter) do not generate any security conflicts;
6. delivery the message to the upper layer;

To rightly complete both aforementioned procedures, each device uses other sub-routines for different purposes. They include: (i) the *KeyDescriptorLookup* procedure, exploited to find the key in the *MacKeyTable*; (ii) the *DeviceDescriptorLookup* procedure, used to find the right entry in the *MacDeviceTable*; (iii) the *SecurityLevelDescriptorLookup* procedure, used to detect the type of frame to be unsecured, (iv) the *IncomingSecurityLevelChecking* procedure, exploited to check if the security level of the incoming frame conforms to the standard security levels for the receiving node; and (v) the *IncomingKeyUsagePolicyChecking*, which identifies if the key is properly used.

To effectively encrypt and decrypt MAC packets, the IEEE 802.15.4 specification imposes the use of the AES-CCM* algorithm.

CCM* is a *wrapper* cryptographic primitive around AES-128 that uses CTR mode for encryption and CBC-MAC for authentication. It encrypts and/or authenticates an arbitrarily long sequence of plaintext bytes (note that a theoretic upper bound on the message length for CCM scheme exists, but for practical purposes it can be considered arbitrarily long for IIoT technology) . When applied to a link-layer frame, this means that CCM* can encrypt the MAC payload while keeping the MAC header intact. When used to authenticate a frame, CCM* calculates a 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. This is depicted in Fig. 2.3.

The *CCM* mode forward procedure*, adopted to protect a MAC message, involves the execution of three subroutines:

1. the *Input Transformation*, which is in charge of creating two data strings, i.e., *AuthData* and *PlainTextData*, that represent initialization vectors in next steps;

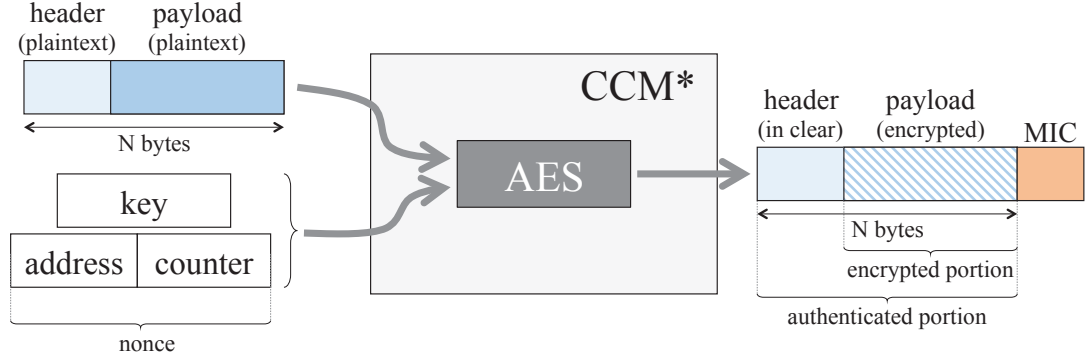


Figure 2.3: CCM* allows every frame to be encrypted and/or authenticated.

2. the *Authentication Transformation* procedure, which generates the authentication tag. It is based on the CBC-MAC mode;
3. the *Encryption Transformation*, that generates the encrypted message through a AES-Counter Mode (CTR) scheme.

Finally the *CCM* mode inverse transformation* handles the decryption and the message integrity checking through the *Decryption Transformation* and the *Authentication Checking Transformation* function, respectively. Without loss of generality, it is possible to note that these functions execute the same subroutines described for the *CCM* mode forward procedure*.

Changes introduced with link-layer security in IEEE802.15.4e-2012. IEEE802.15.4e introduces Information Elements (IE) to exchange information between neighbor nodes in a TSCH network. Nodes maintain synchronization by indicating the time correction as part of an IE in Acknowledgement (ACK) frames. An attacker could perform DoS attacks by altering this time correction; for this reason IEEE802.15.4e-2012 added support for secured ACK frames. Time synchronization in the network means that all nodes share the ASN: the number of slots which have passed since the network has started. The ASN is forever incrementing¹. 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 Auxiliary Security Header (ASH), reducing the overhead by 4 bytes.

¹The ASN is typically encoded on 5 bytes. With a 10 ms timeslot duration, the ASN value rolls over every 350 years.

2. SECURING IOT SYSTEMS

These novel functionalities bring to important issues in MAC-layer security operations. These aspects will be investigated in Section 3.

Further amendments introduced with IEEE802.15.4-2015. The security sub-layer standardized in IEEE802.15.4-2015 [22] generally integrates security services defined in both IEEE802.15.4-2011 and IEEE802.15.4e-2012. However, some few amendments have been introduced.

First of all, the ENC security level is not supported anymore because it is not robust against trivial Man-In-The-Middle attacks. Secured and unsecured communications can coexist in the same network (for instance, in the case a device has not enough resources to manage security, it can use a *Capability IE* to inform its neighbors about this issue, thus asking them to switch to unsecured communications). Moreover, IEs and specific frames introduced by previous specifications (such as command frames defined in IEEE802.15.4e-2012) can be protected.

A different management of the frame counter is defined for the two possible network operation modes. In the legacy mode, it is an integer value associated to the device and to the key used to protect the communication. In the TSCH mode, instead, the frame counter is elided from the packet.

Few updates of both ongoing and incoming security functionalities have been added in order to take care of aforementioned amendments. Nevertheless, these changes do not modify the computational overhead introduced by security functionalities defined in IEEE802.15.4e-2012, which is instead mainly due to AES and CCM* operations.

2.6.2 Layer-2 security: literature review

In what follows it is provided a detailed overview of the main topics and related activities around security in IEEE 802.15.4. The focus will be on both standardization activities and efforts provided by the research community.

2.6.2.1 Link-layer security overhead

Around IEEE 802.15.4, a lot of work have been done in order to evaluate the overhead introduced by security operations and, especially, the AES-CCM operation.

The AES-CCM scheme can be used at different layers of the protocol stack. For instance, in [55] it is used for application-layer security. [56], [57] and [58] further discuss its suitability for the TLS/DTLS protocols at the transport layer. In [59] and

2.6 State of the art on security protocols for IoT networks

[60] it is used at the network layer of an IoT-compliant protocol stack. Finally, in [61, 62] it is used at the link layer.

In these last works, the authors show, by experimentation, that the hardware support provided by different platforms can be used to significantly speed up the execution of cryptographic primitives.

[63] studies the performance of the AES-CCM at the link layer, and compares it to other well-known cryptography schemes. The authors show that, even though AES is not the most efficient block cipher in terms of code-size and time performance, its well-known security properties, as well as the possibility to perform it in hardware, makes it the most suitable solution. The work presented in [64] evaluates the impact that cryptographic algorithms have on IoT resources (i.e. memory, processing, energy). It demonstrates that the Rijndael algorithm, which is the AES scheme, is one of efficient ciphers for IoT devices (as other block ciphers), offering a good compromise between code-size and cycle-count.

Several works discuss the advantages of hardware acceleration for security operations. The first contribution that highlighted the performance gain offered by hardware accelerators in CC2420-based platform (i.e. Crossbow MICAz and MoteIV's TmoteSKY) was [65]. In [66], the authors describe a compact and energy-efficient hardware implementation of IEEE802.15.4 security, and show its advantage in terms of execution speed and energy consumption. In [67], the authors design an energy-efficient hardware architecture for AES-CCM for IEEE802.15.4 networks. [68] evaluates hardware, software, and hybrid implementations of the AES encryption engine on IoT micro-controllers, along with a discussion about the trade-offs between energy, throughput of the hardware module, memory footprint, and battery lifetime. Nevertheless, authors do not discuss the integration of AES cryptography algorithm with IEEE 802.15.4(e) security processing, as well as the impact derived by the adoption of these solutions on the overall network operation.

Similar to the current article, [63, 69, 70, 71, 72] analyze the impact of IEEE802.15.4 security processing on network performance. [69] analyzes the impact of the inclusion of link-layer encryption and authentication services on energy and memory consumption, on Tmote-Sky motes. However, only the hardware implementation is presented, timing issues are not considered, nor is the integration with TSCH. [70] analyzes the impact of link-layer security processing on IEEE802.15.4 network performance (at the

2. SECURING IOT SYSTEMS

application layer) and on other protocol parameters. The authors focus on time and energy consumption. By simulations, they show that encryption and authentication affect the end-to-end application delay, but they do not include a discussion about the impact of link-layer security on the performance of the MAC protocol. [63] presents an implementation and performance evaluation of security functionalities at the link layer of IEEE802.15.4-compliant IoT devices. The authors present software and hardware implementations of security functionalities over the AVR XMEGA platform, evaluating memory and energy consumption produced by each one. Authors in [73] propose a novel mathematical tool for analyzing the IEEE 802.15.4 network performance, considering a fixed slot duration. The included model is perfectly integrable in our work, and can be used to derive different MAC delays when security is enabled. Authors in [71] evaluate the energetic cost of IEEE802.15.4 security in the context of energy harvesting devices and beacon-enabled mode. However, the impact of security processing over the TSCH mode of IEEE802.15.4e-2012 is not considered. An analytical evaluation of security overhead in the IEEE802.15.4 networks only is also presented in [72].

Several works focus on the new IEEE802.15.4e-2012 TSCH standard [17, 74, 75].

Yet, to the best of the author knowledge, no work focuses on the performance of link-layer security. These aspect will be further investigated in Section 3

2.6.2.2 Standardization Activities

From the standardization perspective, the IETF 6tisch WG is particularly active on IEEE 802.15.4 security-related topics [76]. It targets the definition of: (i) the keying material and authentication mechanism needed by a new node to join an existing network, (ii) a mechanism to allow the secure transfer of application data between neighbor nodes, (iii) a scheme to secure signaling data. In this context, two relevant Internet Drafts have been published. All the relevant elements for the design of the 6tisch security architecture are discussed in [77]¹. Instead, the description of a more complete security architecture for industrial environments, which covers minimal security features for both layer-2 and layer-4, is investigated in [78]. In particular, [78] proposes to adopt DTLS-based solutions for enabling secure communications at the application layer.

¹[77] expired and a new proposal was not presented.

Finally, a mechanism to provide each joining node with a dedicated session key is being currently investigated by the 6tisch security team in [32].

2.6.2.3 Key Management Protocols

One of the gaps left open by the IEEE 802.15.4 standards is the negotiation of a key between any couple of neighbors. The initialization of a secure link and the negotiation of symmetric keys is generally handled through a Key Management Protocol (KMP). In this context, either *centralized* or *distributed* solutions can be used.

Centralized approaches, like those presented in [79]-[81], suppose the use of a dedicated entity for generating, distributing, and managing keys. In distributed techniques, instead, each couple of nodes negotiate key materials autonomously, as discussed in [82]-[88]. In particular, in [84], [83] and [85], the link key is calculated by using innovative techniques based on channel measurements. KMP schemes presented in [82] and [88] create a *group key* (also known as *cluster key*), which is shared into the network for different purposes (e.g., node authentication and message encryption). In [82], link keys are generated by means of pseudo-random functions, which receive as input variables a shared secret, i.e., the master key, and other common parameters characterizing the clustering structure of the Wireless Sensor Networks. Note that distributed mechanisms presented in [82]-[85], [88], and presented before, are based on symmetric key encryption schemes. Further distributed approaches use asymmetric strategies, based on the usage of public key cryptography [89][79][88][90].

To conclude, it is worth to note that a new standardization initiative, namely the IEEE 802.15.9 standard [91], is currently addressing the support for KMP messages through IEEE 802.15.4 Information Elements. It includes the main Internet protocols, such as Extensible Authentication Protocol (EAP), Internet Key Exchange (IKE)v2 and Protocol for carrying Authentication for Network Access (PANA), but it does not completely solve the problem, not proposing any of these mechanisms as the final or preferred solution.

2.6.3 Application-layer security in IoT networks

At the application layer, the most important works focus on authentication, key management and access control.

2. SECURING IOT SYSTEMS

Regarding authentication and key management, the CoAP protocol has modified the widely employed DTLS so as to fit it into memory/energy constrained devices [56] [92].

However, using DTLS as it is does not represent a feasible option. In fact, the DTLS Handshake protocol used for as the KMP at the application layer requires several message exchanges, thus leading to strong overhead, an increase of bandwidth requirements and further energy consumption. For these reasons, a common approach refers to the design of a simplified version of the DTLS and its recommended Handshake protocol. Interesting contribution in this context have been discussed in [93] and [94].

As regarding access control, it is usually enforced at the gateway of an IoT network, thus relieving constrained devices from managing complex messages, databases or Access Control Lists (ACLs).

The OAuth 2.0 authorization framework, widely used in today's Internet, is currently the most attractive and feasible solution to handle authorization [95].

OAuth 2.0 allows a third-party application to obtain the access to a HTTP service on behalf of a resource owner, by offering a mechanism through which demonstrating that this possibility is obtained as a result of an approval interaction between the resource owner and the HTTP service. Access grants are released on a token-based approach. But, the definition of tokens is out-of-scope for the reference specification.

Moreover, thanks to its flexibility and decentralized design, OAuth 2.0 seems emerging as a successful mechanism for handling access control also in the IoT.

The IETF Authentication and Authorization for Constrained Environments (ACE) working group, for instance, is promoting the adoption of the OAuth 2.0 approach in scenarios where both applications and resources are available on constrained devices. To this end, OAuth 2.0 instructions are re-defined in order to be delivered through CoAP messages [96]. Moreover, new token formats, more suitable for low-power short-range and lossy wireless links, are introduced [97].

At the same time, the research community is also moving the attention to a different case where a third-party application is not within the constrained IoT environment anymore, but it can be remotely attached to Internet. A preliminary high-level design of the reference architecture was discussed in [98]. A concrete proposal, instead, is discussed in [99]. Specifically, it considers the possibility to expose IoT resources through both HTTP-based and CoAP-based servers and adopts OAuth 2.0 primitives

2.6 State of the art on security protocols for IoT networks

for managing authorization services. Furthermore, it introduced a new centralized entity (namely IoT-OAS) that is in charge of validating access requests sent by a third-party application, on behalf of the server that exposes IoT resources. Unfortunately, this additional component makes the proposal not completely standard compliant and reaches the integration of OAuth 2.0 and accepted communication protocols for the IoT by means of a customized scheme.

2. SECURING IOT SYSTEMS

3

Advanced MAC-layer security functionalities for large-scale IoT networks

The discussion carried out in Chapter 2 highlighted some important gaps that need to be filled in the context of IoT security. This chapter focuses on MAC layer security, specifically on IoT networks based on the IEEE 802.15.4 standard. These networks at the MAC layer must follow procedures and mechanisms defined in the respective technology specification, IEEE 802.15.4 [22]. The OpenWSN protocol stack, that implements the protocol suite defined by the IETF WG 6tisch, in its release 1.9.0 does not include any of the security routines defined by the standard. Therefore, this chapter firstly describes the work needed to include the layer-2 security procedures within this protocol stack (see Section 3.1). The discussion covers not only the legacy IEEE 802.15.4-2011 standard, but also the TSCH mode defined in IEEE 802.15.4e-2012 and further included in the last version IEEE 802.15.4-2015, and compares different implementations of the AES-CCM algorithm and different experimental boards (see Section 3.2).

Moreover, recall from the discussion in Section 2.6.1 that the IEEE 802.15.4 standard does not define any mechanism for initializing network security and negotiating session keys between peers. To this aim, this chapter describes *LICITUS*, a lightweight and standard compatible framework aimed at securing layer-2 communications in the IoT, through several functionalities (see Section 3.3).

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

As a concrete result of the above research efforts, the *TLSensing* testbed, implementing link-layer security techniques and procedures described throughout the rest of the section, is showed in Section 3.4 and described along with some of its possible applications.

The chapter concludes with the description of an innovative KMP working at the physical layer, that allows two directly connected peers to establish a shared secret without the use of any cryptography material, by simply grounding its basis on the wireless channel intrinsic anonymity.

3.1 Implementation of IEEE 802.15.4 security procedures in OpenWSN protocol stack

At the time of this writing, one of the most promising free-ware and open-source implementation of IoT-compliant protocol stack for constrained devices has been developed within the OpenWSN project [44]. Nevertheless, despite it already offers a large number of protocols, including PHY and MAC layers defined in IEEE 802.15.4 and IEEE 802.15.4e specifications, as well as other high level protocols devised by the IETF, i.e., 6LoWPAN, RPL, and CoAP, some important features are still not fully available.

A comprehensive scheme of the OpenWSN protocol stack in its version 1.9.0 has been reported in Fig. 3.1.

Among them, one of the most important lacks is the unavailability of security procedures and services defined by the IEEE 802.15.4 standard for the MAC layer, as described in Section 2.6.1 [25].

The present contribution intends to overcome this weakness by proposing an open source implementation of IEEE 802.15.4 security features within the OpenWSN project. To this end, it properly extends the OpenWSN protocol stack by developing:

- MAC PIB attributes related to security aspects, i.e., tables and variables storing all the information needed to provide security services (including keys, devices' authorization, security levels, and so on);
- functions operating at the MAC-high layer, handling the inserting and the retrieving of security-related parameters and attributes introduced at the previous point;

3.1 Implementation of IEEE 802.15.4 security procedures in OpenWSN protocol stack

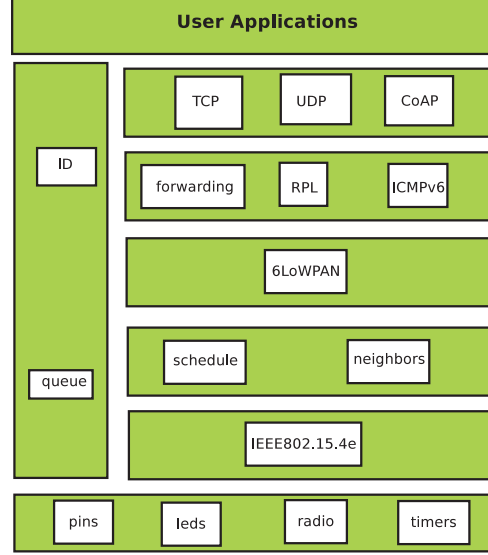


Figure 3.1: The OpenWSN default protocol stack.

- basic cryptographic techniques used to execute encryption and decryption functionalities, based on a software version of the AES scheme.

It is worth remarking that the developed code is fully compliant with IEEE 802.15.4 specifications and it is integrated with the entire OpenWSN protocol stack (and, hence, it could be used for managing security services at the MAC layer independently from the set of configured upper layers). Moreover, the code is freely available at: <http://telematics.poliba.it/openwsn-ieee802154-security>.

3.1.1 Implementation details

In order to overcome the aforementioned lack, a proper extension of the original version of the OpenWSN protocol suite has been designed, by developing (i) MAC PIB attributes related to security aspects (ii) security functions operating at the MAC-high layer, and (iii) cryptographic techniques. A clear description of entities forming the developed module, including their relationship with the existing OpenWSN protocol stack, is depicted in Fig. 3.2.

First of all, in line with the IEEE 802.15.4 standard, a number of data structures and variables, which refer to MAC PIB security attributes, have been developed. The most important implementation details can be found in Tab. 3.1. In addition, to enable

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

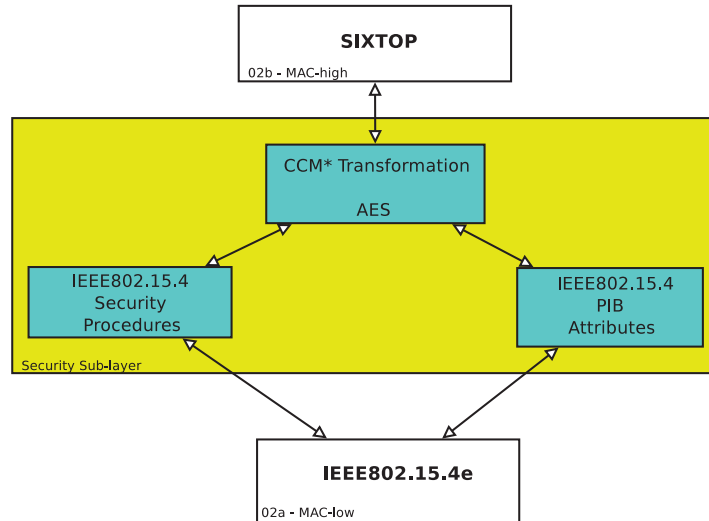


Figure 3.2: The OpenWSN modified portion of the stack.

security aspects, the *OpenQueueEntry.t* structure, which models the data packet, has been extended by adding the following variables:

- *l2_security*, reporting if security features are enabled or not for that packet (data type = *boolean*);
- *l2_securityLevel*, which specifies the security level of the frame (data type = *uint8_t*);
- *l2_keyIdMode*, indicating the *KeyIdMode* value to store into the MAC header (data type = *uint8_t*);
- *l2_keyIndex*, indicating the *KeyIndex* value to store into the MAC header (data type = *uint8_t*);
- *l2_frameCounter*, storing the frame counter (data type = *uint32_t*);
- *l2_keySource*, indicating the *KeySource* value to store into the MAC header (data type = *open_addr_t*);
- *l2_authenticationLength*, storing the length of the authentication field to append to the packet (data type = *uint8_t*);

3.1 Implementation of IEEE 802.15.4 security procedures in OpenWSN protocol stack

Implemented Parameter	Description	Composition
<i>m_deviceDescriptor</i>	structure modeling the <i>DeviceDescriptor</i> attribute	device address (data type = <i>open_addr_t</i>), Frame Counter (data type = <i>integer</i>), Exempt (data type = <i>boolean</i>)
<i>m_keyIdLookupDescriptor</i>	structure modeling the <i>KeyIdLookupDescriptor</i> attribute	Key Identifier Mode (data type = <i>uint8_t</i>), Key Index (data type = <i>uint8_t</i>), Key Source (data type = <i>open_addr_t</i>), PAN ID (data type = <i>open_addr_t</i>) and device address (data type = <i>open_addr_t</i>)
<i>m_securityLevelDescriptor</i>	table modeling the <i>SecurityLevelDescriptor</i> attribute	<i>DeviceOverrideSecurityMinimum</i> (data type = <i>boolean</i>), Frame Type (data type = <i>uint8_t</i>), Command Frame Identifier (data type = <i>uint8_t</i>), Security Minimum (data type = <i>uint8_t</i>), and Allowed Security Levels (data type = <i>uint8_t</i>)
<i>m_keyUsageDescriptor</i>	structure modeling the <i>KeyUsageDescriptor</i> table	<i>FrameType</i> (data type = <i>uint8_t</i>) and <i>CommandFrameIdentifier</i> (data type = <i>uint8_t</i>)
<i>m_macDeviceTable</i>	table modeling the <i>MacDeviceTable</i> attribute	vector of <i>m_deviceDescriptor</i> elements
<i>m_keyDescriptor</i>	structure modeling the <i>KeyDescriptor</i> attribute	<i>m_keyIdLookupDescriptor</i> element, <i>DeviceDescriptorHandleList</i> (i.e., pointer to the <i>m_macDeviceTable</i>), <i>m_keyUsageDescriptor</i> element, and key (data type = vector of <i>uint8_t</i>)
<i>m_macKeyTable</i>	table modeling the <i>MacKeyTable</i> attribute	vector of <i>m_keyDescriptor</i> , elements
<i>m_macSecurityLevelTable</i>	table modeling the <i>MacSecurityLevelTable</i>	vector of <i>m_securityLevelDescriptor</i> elements
<i>m_macFrameCounter</i>	variable modeling the <i>MacFrameCounter</i>	single variable (data type = <i>uint32_t</i>)
<i>m_macDefaultKeySource</i>	variable modeling the <i>MacDefaultKeySource</i>	single variable (data type = <i>open_addr_t</i>)

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

Table 3.1: Description of IEEE 802.15.4 MAC security attributes implemented in OpenWSN.

To enable security features at the MAC layer, *Outgoing Frame Security*, *Incoming Frame Security*, *Key Descriptor Lookup*, *Device Descriptor Lookup*, *Security Level Descriptor*, *Incoming Security Level Checking*, and *Incoming Key Usage Policy Checking* procedures have been implemented as imposed by the IEEE 802.15.4 standard and described in sec. 2.6.1. They are handled at the MAC-high layer of the protocol stack (in particular within the *SIXTOP* module, as depicted in Fig. 3.2), and they are also in charge to build the *Auxiliary Security Header* and retrieve data from it (see Sec. 2.6.1 for more details). Encryption and decryption operations have been implemented in *Input Transformation*, *Authentication Transformation*, *Encryption Transformation*, *Decryption Transformation*, and *Authentication Checking Transformation* functions.

Finally, with respect to cryptographic primitives, a free and open-source implementation of the AES-128 algorithm¹ has been initially included, too. The evaluation of performances of such a module will be described thoroughly in Section 3.2.

3.1.2 Usage of the developed module

The developed module is perfectly integrated within the overall OpenWSN protocol stack.

When the *SIXTOP* module of the MAC-high layer receives from upper layers a packet with the *l2_security* flag set to TRUE, it invokes, as described in Section 2.6.1, the implemented *OutgoingFrameSecurity* procedure by providing the *l2_securityLevel*, the *l2_keySource*, the *l2_keyIndex* and the Next Hop address.

In the case the message is received from the physical interface and the *l2_security* flag is set to TRUE, instead, the *taskSixtop_NotifReceive* function (that is called during the decapsulation process) delivers the message to the *IncomingFrameSecurity* procedure.

Finally, the initialization of MAC PIB attributes reported in Tab. 3.1 should be statically handled by the user by setting their values in the *security_init* function, which is executed into the OpenWSN stack boot-up process (i.e., *OpenWSN_init* function).

¹It is available at <http://comp.ist.utl.pt/ec-csc/Code/Ciphers/AES.Encrypt.cpp>

3.2 Link-layer security in TSCH networks

The implementation of security procedures and mechanisms described in the Section 3.1 represents the basis for a new interesting research activity, developed by moving the focus on the TSCH mode of the IEEE 802.15.4 standard.

In fact, as emerged from the discussion in Section 3.1, securing and unsecuring frames are complex and potentially time-consuming operations. Moreover, different platforms have different hardware capabilities: on some, link-layer security can be accomplished using hardware support; on others, these operations need to be done in software because of the lack of hardware accelerators. Anyway, when security procedures are integrated in the TSCH communication schedule, a further computational overhead is introduced and the (minimum) time-slot duration is expected to increase as well, due to the additional time needed to secure/unsecure frames [5].

At the time of this writing, several contributions already investigated the effects of security operations in IoT systems on system capabilities [63, 69, 70, 71, 72]. Nevertheless, they only focus on the legacy IEEE802.15.4-2011 [24] and no studies on IIoT networks based on TSCH are yet available (TSCH was not present in the legacy IEEE802.15.4-2011 standard, while it was introduced in the IEEE802.15.4e amendment, in 2012, and maintained in the latest version of the standard, that is IEEE802.15.4-2015 [22]).

To bridge this gap, this section aims at experimentally measuring the overhead of link-layer security in TSCH networks, on the minimal duration of a time-slot and on the memory footprint of the implementation [5].

It is important to remark that this is not a proposal of a new link-layer security mechanism. On the contrary, it represents a deep investigation on the impact that the one already proposed by the IEEE802.15.4e amendment, and maintained in IEEE802.15.4-2015, has in real IIoT systems. The scientific contribution provided by such a research activity is four-folded:

- It discusses the security functionalities in TSCH networks, the communication schedule described in the IEEE802.15.4e amendment (and maintained in IEEE802.15.4-2015), and the hardware support for link-layer security operations in today's IIoT platforms.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

- It provides a full implementation of link-layer security on older and state-of-the-art platforms, using different hardware/software strategies. The produced open-source code is released under the BSD license¹.
- It includes an exhaustive experimental study and measure that the minimal value of the time-slot duration is between 9 ms and 88 ms (for the most common security level, described in Section 2.6.1), across the different platforms and strategies.
- It describes the impact that the minimum time-slot has on both high-level application design and energy consumption [5].

3.2.1 Time-slot duration in the TSCH mode

Fig. 3.3 depicts the operations that happen in a timeslot when A sends a data frame to B:

- A secures the data frame. This operation is called *sec₁*.
- 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, which can involve decrypting and/or authenticating it (operation *sec₂*).
- If the unsecuring operation is successful, B secures an ACK frame (operation *sec₃*).
- 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 *sec₄*). If successful, it removes the data frame from its transmission queue.

¹ http://telematics.poliba.it/openwsn_ieee802154_security

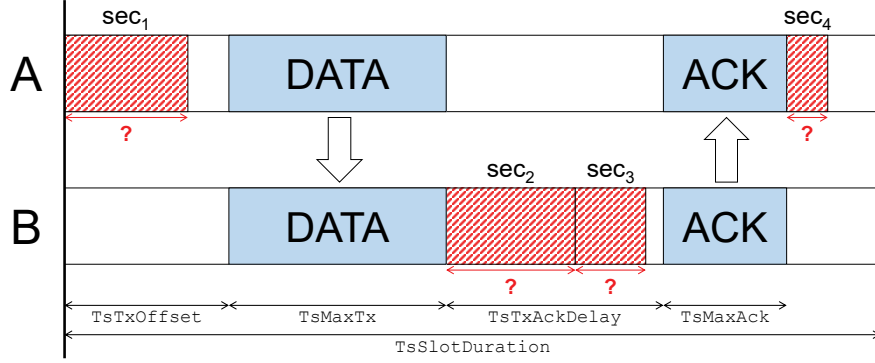


Figure 3.3: The minimal duration of a timeslot in an IEEE802.15.4e TSCH network depends on how long link-layer security operations take.

Eq. (3.1) indicates the timing constraints that the duration of those security operations put on the different TSCH timings. $dur(sec_1)$ denotes the duration of sec_1 .

$$\begin{cases} TsTxOffset \geq & dur(sec_1) \\ TsTxAckDelay \geq & dur(sec_2) + dur(sec_3) \\ TsSlotDuration \geq & TsTxOffset + TsMaxTx + \\ & TsTxAckDelay + TsMaxAck + \\ & dur(sec_4). \end{cases} \quad (3.1)$$

The TSCH mode uses the ASN as a frame counter and omits its inclusion in the ASH. From one side, this reduces the overhead by 4 bytes. From the other side, the ASN is used also in the construction of the nonce string, used for the CCM* cryptography operation. 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. 3.3). 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 transmission side (node A in Fig. 3.3), 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.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

Table 3.2: Implementation Strategies.

strategy		AES	CCM*
“software”	(Sec. 3.2.3)	software	software
“hybrid”	(Sec. 3.2.5)	hardware	software
“hardware”	(Sec. 3.2.4)	hardware	hardware

Moreover, also the mode in which the CCM operation is executed influences the time-slot duration. It can be implemented by using a software strategy, a full-hardware technique or an hybrid approach.

The three different implementations strategies are hereby presented: software (Section 3.2.3), hardware (Section 3.2.4) and hybrid (Section 3.2.5).

As shown in Table 3.2, these strategies differ in whether they exploit hardware acceleration for AES and/or CCM*.

All source code developed is available open-source, under a BSD license. ¹.

3.2.2 The hardware support for security in TelosB and OpenMote-CC2538

Both TelosB and OpenMote-CC2538 support the possibility to execute the AES algorithm or the CCM operation directly in the hardware, thus reducing the time needed to complete these potentially time-consuming operations.

The CC2420 radio [100] chip included in the TelosB mote provides hardware acceleration for both AES and CCM*.

To execute an AES operation in hardware on the CC2420, one loads 16 bytes of clear-text into a buffer and a 16-byte key in a dedicated register, and issues a **SAES** command. This “atomic” AES operation takes $14\mu s$, a number which does *not* account for the time to load the buffer and key from the micro-controller into the radio chip. Also, one has to execute this operation for each 16-byte block of a long frame.

To execute a CCM* operation in hardware on the CC2420, one sets the **Security Control 0** and **Security Control 1** registers to configure the security mode, the length of the authentication tag, the length of data to be authenticated but not encrypted, and the exact position where authenticated data starts. The nonce has to be

¹ http://telematics.poliba.it/openwsn_ieee802154_security

loaded in the `TXNONCE` register and the frame into the transmission queue. Issuing a `STXENC` executes a CCM operation in the buffer; issuing `STXON` additionally transmits the frame. Considering a message of 119 bytes, a CCM operation takes $222\mu s$.

The TelosB is a two chip-solution: the micro-controller and radio chips are connected by an SPI digital interface. Given the speed of this bus, it takes approximately 1 ms to transfer a full-sized frame between the chips. This adds a extra delay to the minimal timeslot duration.

The CC2538 SoC included in the OpenMote platform uses a single `AES Control` register to configure/trigger AES and CCM* operations in hardware. The AES module cuts the input bytes into 16-byte blocks and triggers the AES operation autonomously. When the operation finishes, the master controller generates a `data_done` interrupt so the software can retrieve the encrypted data. The software sets the nonce, key and other CCM* configurations in the `AES Control` register.

A Direct Memory Access (DMA) modules takes care of feeding the AES and CCM* modules, without intervention from the software. This significantly speeds up the operation when compared to the CC2420. Encrypted/authenticated data is available directly in RAM memory, no (time consuming) SPI transfers are needed. Finally, the external 32 MHz crystal oscillator is used to clock the AES and CCM* hardware modules. This is possible because the AES cryptoprocessor can be clocked independently from the processor.

The AES algorithm executes at 18 Mbps¹. CCM executes at 12 Mbps, including the final operation of create the MIC. Thus, considering a single block of 128 bits, security operations are executed in $7\mu s$ for the AES algorithm and $85\mu s$ for the CCM scheme. However, processing multiple blocks increases performances significantly. The CC2538 completes AES and CCM tasks faster than the CC2420 [5].

3.2.3 Software Implementation

The *software* implementation strategy consists in implementing both AES and CCM* in software. A software AES implementation from Texas Instruments² has been used, unmodified. This implementation is optimized for embedded devices, and features an execution time approximately 10 times shorter than the baseline AES implementation

¹<http://www.ti.com.cn/cn/lit/ug/swru319c/swru319c.pdf>

² <http://www.ti.com/tool/AES-128>

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

presented in [25]. CCM*, instead, has been implemented from scratch. As indicated above, all software is available open-source.

It is worth noting that in the software implementation, the hardware acceleration for AES/CCM* offered by both platforms has purposely *not* been used, in order to be able to (1) quantify how much shorter the timeslot can be when using hardware acceleration, and (2) have an idea of the performance of a platform which does not have hardware acceleration [5].

3.2.4 Hardware Implementation

The *hardware* implementation strategy consists in exploiting hardware acceleration for both AES and CCM*. Both platforms offer AES and CCM* hardware support; therefore, it is possible to use the techniques described in Section 3.2.2 in the implementation.

Note that even though the hardware executes the core of the security operation, software instructions are necessary to store security-related parameters in the correct location, disable or enable hardware interrupts, give the “go” signal for encryption/decryption operations, and fetch the output of the operation [5].

3.2.5 Hybrid Implementation

Many hardware platforms offer AES hardware acceleration, but not CCM*. To measure the overhead of link-layer security on those platforms, an *hybrid* implementation strategy has been adopted, in which AES is implemented in hardware, but 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 [5].

3.2.6 Performance Evaluation

This section presents the experimental results obtained on the implementations presented in Sections 3.2.3, 3.2.4 and 3.2.5. Section 3.2.6.1 details the experimental setup and the methodology adopted to tune the different durations in the time-slot. Section 3.2.6.2 indicates the memory footprint of the implementations, for both flash and

RAM memory, in the OpenWSN protocol stack. Section 3.2.6.3 presents the different durations measured on both platforms, across the three implementation strategies, and for all eight security levels. It includes also the discussion about the implication for IEEE802.15.4e TSCH networks. Section 3.2.6.4 discusses the energy consumption related to the three implementation strategies, and for all eight security levels. Finally, Section 3.2.6.5 provides further comments on the impact of timeslot duration on high-level application design.

3.2.6.1 Goals and Methodology

The goal of the conducted experimental evaluation is to implement the three implementation strategies on both platforms, tune the time-slot duration to the minimal value obtainable given the duration of the link-layer security operations, and measure the different timings. Specifically, the experimental evaluation wants to answer the following question: *How much shorter can a time-slot be if hardware acceleration is used?*

The experimental setup consists of two nodes forming one network. 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, the root node is booted and the leaf node synchronizes to it. The root node is attached to a computer; from that computer the `ping` program is used to verify connectivity to the leaf node. `ping` allows to choose the size of the payload sent in the ICMPv6 echo request/response packets; it has been chosen so that the resulting link-layer frame is always 127 bytes long (the maximum length for IEEE802.15.4-compliant nodes).

In OpenWSN, IEEE802.15.4e TSCH is implemented as a finite state machine. Different timings, illustrated in Fig. 3.3, 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 time-slot, all the operations for preparing the packet (including `sec1`) need to be complete. The OpenWSN implementation uses a 32-kHz counter to measure the duration of different phases of the IEEE 802.15.4e finite state machine. This allows to measure the duration of the `sec1`, `sec2`, `sec3` and `sec4` operations, and verify the `TsTxOffset`, `TsTxAckDelay` and `TsSlotDuration` durations.

For each evaluated case, the different durations has been tuned in order to obtain the shortest possible time-slot. The procedure starts by setting `TsTxOffset`, `TsTxAckDelay`

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

Table 3.3: The different steps in the tuning procedure.

$dur(sec_1)$	measure
TsTxOffset	tune to $dur(sec_1)$
TsMaxTx	set to 4.256 ms
$dur(sec_2)$	measure
$dur(sec_3)$	measure
TsTxAckDelay	tune to $dur(sec_2) + dur(sec_3)$
TsMaxAck	set to 2.400 ms
$dur(sec_4)$	measure
TsSlotDuration	tune to $TsTxOffset + TsMaxTx + TsTxAckDelay + TsMaxAck + dur(sec_4)$

and **TsSlotDuration** to very high values, then it goes ahead by progressively reducing them. In all cases, these values must satisfy the constraints in Eq. (3.1). The different steps of the tuning procedure are illustrated in Table 3.3. The duration of sec_1 has been measured firstly, and the value of **TsTxOffset** have been reduced accordingly. According to the standard [21], **TsMaxTx** and **TsMaxAck** are set to 4.256 and 2.400 ms respectively. Then the durations of sec_2 and sec_3 have been measured, and the value of **TsTxAckDelay** have been set accordingly. Similar technique has been applied for measuring the duration of sec_4 and tuning **TsSlotDuration**.

This tuning operation have been repeated for both platforms, using the three implementation approaches, and for each of the 8 security levels. There are hence 48 cases, discussed in the following sections.

Note that sec_1 , sec_2 , sec_3 , and sec_4 have been measured by using the 32KHz counter available in the OpenWSN protocol stack. Now, considering that the variability of the amount of time needed to execute security operations is very lower than the resolution of this counter, obtained results (reported in the following subsections) appear as deterministic.

Finally, results related to the energy consumption are generated by considering the amount of energy consumed by TelosB and OpenMote-CC2538 platforms when they perform different operations in a time-slot (reference values are taken from their datasheets).

3.2.6.2 Memory Footprint

The memory footprint is the amount of flash ¹ and RAM memory that needs to be reserved for link-layer security. Fig. 3.4 and 3.5 summarize the results. The security level does not impact the memory footprint, and is hence not presented. Since the TelosB and OpenMote-CC2538 have both different processor architectures and compilers, and since different toolchains are used to build their software, comparing the absolute value of the memory footprint is not appropriate. Focusing on a single platform, instead, the analysis of both ROM and RAM footprints is useful to compare different link-layer security implementations.

On the TelosB mote, link-layer security occupies between 7% and 15% of the available flash memory, and 30-40% of RAM memory. On the OpenMote-CC2538, it occupies 1.5-2.5% of flash and 8% of RAM. As can be expected, the footprint of the software (resp. hardware) implementation is the largest (resp. lowest).

In general, it is possible to observe that using hardware acceleration for link-layer security does not result in a drastically lower memory footprint. This is because (i) a software implementation of AES and CCM is relatively simple and (ii) even when hardware acceleration is used, software is still needed to drive the hardware.

On the TelosB mote, the number of instructions needed to control AES and CCM algorithms in hardware is the same. Therefore, the hardware implementation registers a reduction of the ROM footprint with respect to the hybrid implementation because part of the software related to CCM operations (e.g., instructions for CCM* vectors handling) is deleted.

On the OpenMote-CC2538, instead, the driver needed to control CCM operations in hardware has a ROM footprint that is quite similar to the amount of software used in the hybrid implementation. Therefore, this platform seems to report the same ROM footprint when both hybrid and hardware implementations are used.

3.2.6.3 Minimal Slot Length

Figs. 3.6 and 3.7 present the resulting `TsTxOffset`, `TsTxAckDelay` and `TsSlotDuration` durations after tuning procedure, for the TelosB and OpenMote-CC2538 platforms, respectively.

¹The terms flash and ROM (Read-Only Memory) are used interchangeably.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

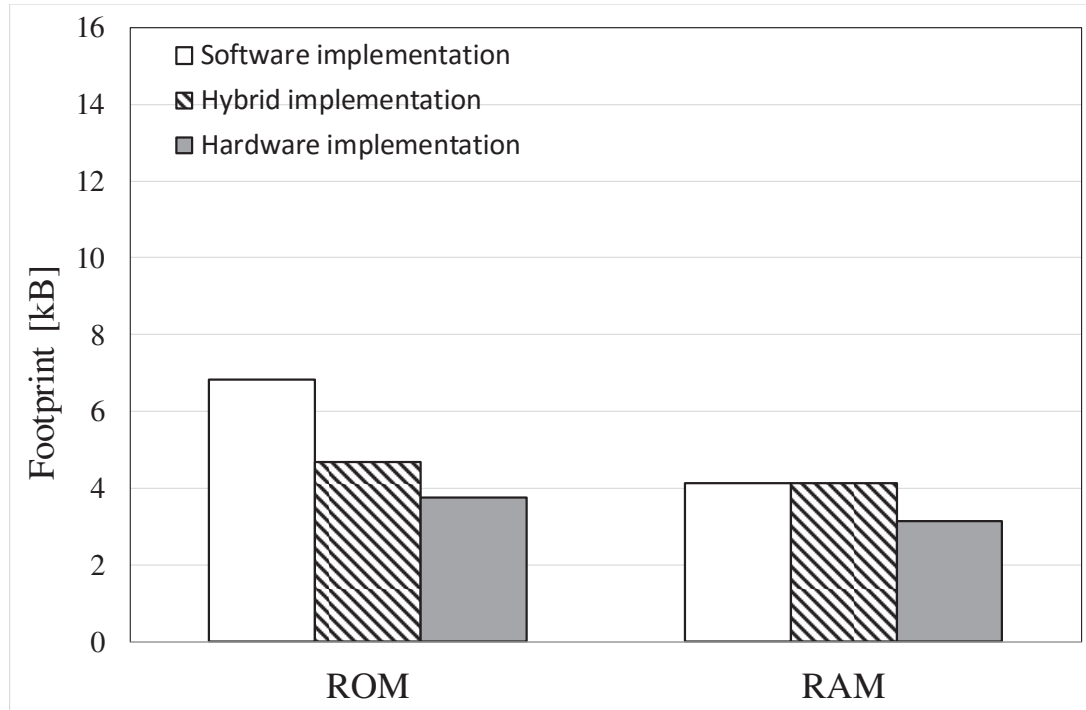


Figure 3.4: ROM and RAM Footprint of the implementations for TelosB mote.

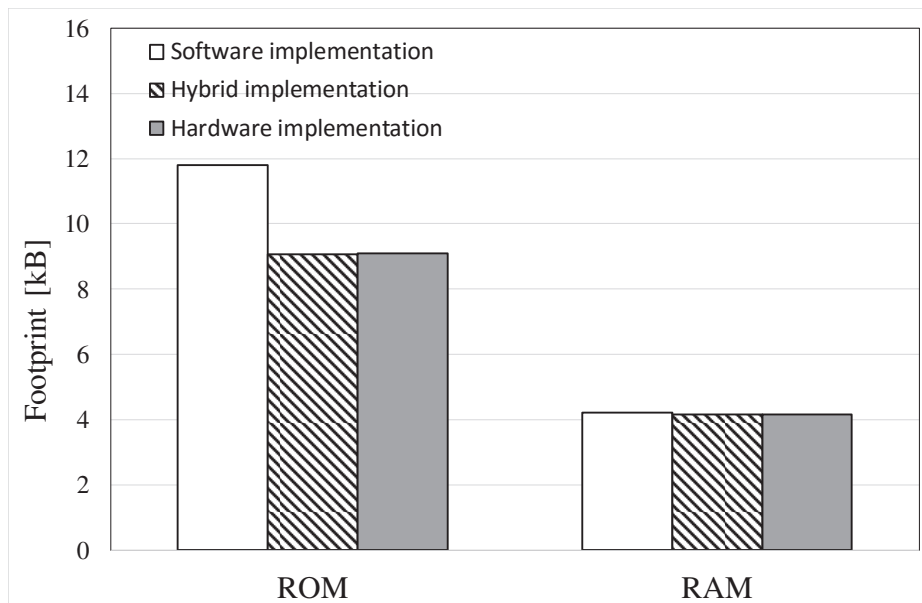


Figure 3.5: ROM and RAM Footprint of the implementations for OpenMote-CC2538 mote.

When the “software” implementation is used, TelosB and OpenMote-CC2538 executes security operations at different speeds of the CPU. For instance, TelosB runs at 8 MHz; OpenMote-CC2538 can run at a maximum speed of 32MHz. As a result, the software-based implementation of link-layer security in OpenMote-CC2538 will result in a timeslot duration reduced by a factor of $2 - 3\times$.

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 $3 - 4\times$.

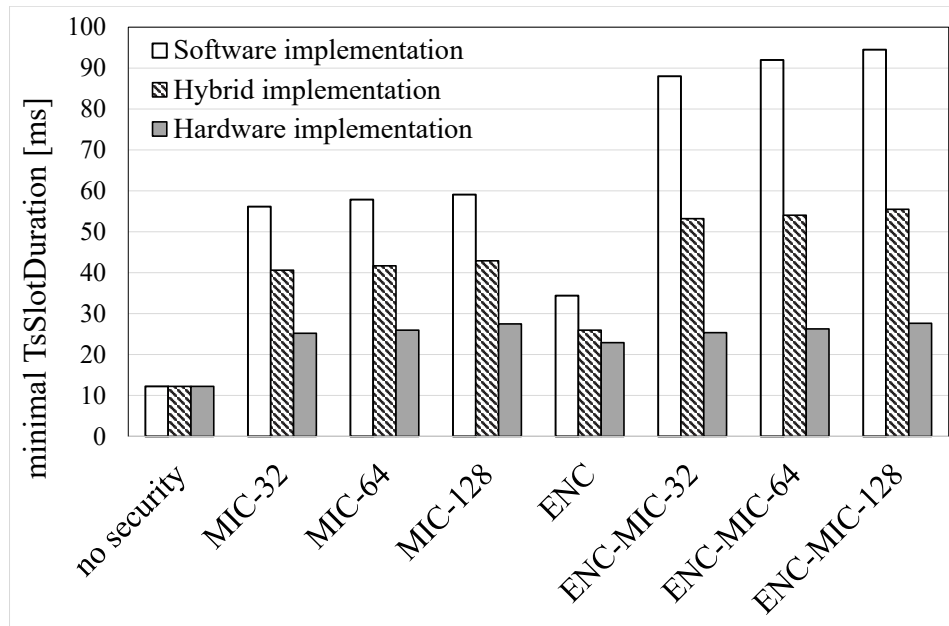
The difference in timeslot duration between “software” and “hybrid” implementation strategies reflects the advantage of having AES execute in hardware. Similarly, the comparison between “hybrid” and “hardware” implementations reflects the advantage of a hardware-based CCM*.

On slower platforms such as the TelosB (Fig. 3.6), the biggest gains are made by running AES in hardware.

The most common security level in TSCH networks (including WirelessHART and 6tisch) is ENC-MIC-32, i.e. frames are encrypted and a 4-byte MIC is used for authentication. That security level has been highlighted in Figs. 3.6 and 3.7. A full software implementation on an older platform such as the TelosB results in a minimal timeslot duration of 88 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 CCM*, and a single-chip architecture, allows the timeslot duration to be further reduce by a factor of more than 2, down to 9 ms.

By using hardware accelerations in OpenMote-CC2538, it is possible to satisfy the target value of the timeslot duration reported by the standard, i.e., 10 ms [21]. In the other cases, this goal is not reached. Anyway, it is important to note that conducted tests considered the worst case, where the maximum size of the ASH is used (e.g. equal to 14 bytes). Now, considering that the higher the size of the ASH, the higher the computational overhead introduced by additional (security) lookup procedures,, all the results reported in Section 5 refer to the most complex network configuration requiring higher timeslot durations. Other configurations, instead, may require lower timeslot durations.

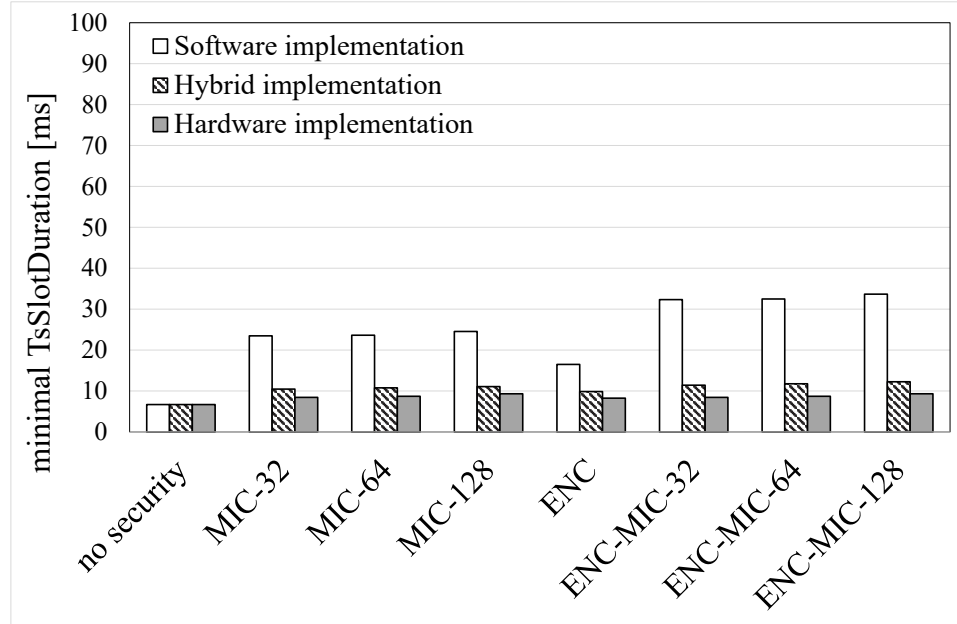
3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS



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	15.71	35.61	56.15	9.90	27.59	40.62	7.75	11.87	25.21
MIC-64	16.32	35.89	57.86	10.11	27.96	41.69	7.81	12.18	25.97
MIC-128	16.72	36.5	59.09	10.42	28.66	42.91	7.90	12.79	27.50
ENC	9.93	20.01	34.38	8.60	11.56	25.96	7.66	10.01	22.92
ENC-MIC-32	26.48	44.22	88.02	14.92	33.24	53.22	7.78	12.48	25.36
ENC-MIC-64	28.25	44.52	91.98	14.98	33.61	54.05	7.83	12.54	26.28
ENC-MIC-128	30.53	45.04	94.49	15.29	34.15	55.51	7.97	12.94	27.65

Figure 3.6: Minimum timeslot duration for the TelosB mote.

3.2 Link-layer security in TSCH networks



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	1.15	0.67	6.71	1.15	0.67	6.71	1.15	0.67	6.71
MIC-32	2.48	10.07	23.50	1.98	2.24	10.47	1.28	1.12	8.45
MIC-64	2.51	10.19	23.65	2.14	2.36	10.77	1.31	1.19	8.72
MIC-128	2.54	10.31	24.57	2.29	2.45	11.08	1.34	1.25	9.34
ENC	2.31	3.20	16.51	1.77	1.51	9.86	1.28	0.98	8.24
ENC-MIC-32	6.44	14.80	32.35	2.38	2.82	11.44	1.28	1.12	8.45
ENC-MIC-64	6.57	14.86	32.50	2.41	2.94	11.78	1.31	1.19	8.72
ENC-MIC-128	6.60	15.26	33.66	2.44	3.24	12.27	1.34	1.25	9.34

Figure 3.7: Minimum timeslot duration for the OpenMote-CC2538 mote.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

3.2.6.4 Energy consumption

Figs. 3.8 and 3.9 reports the amount of energy consumed by TelosB and OpenMote-CC2538, respectively, for the three implementation strategies, and for all eight security levels. These results have been estimated (not experimentally measured) by taking into account the energy consumption related to each phase of the time-slot (TX, RX and CPU idle), as reported by reference data-sheets.

Link-layer security brings to an increment of the per-slot energy consumption because: (i) additional operations are executed for protecting the transmitted packet and decrypting the corresponding ACK and (ii) the radio chip is in charge of transmitting more data due to the ASH and the MIC.

On the TelosB mote, hardware accelerators reduce energy consumption up to 27% (for the hybrid implementation) and 47% (for the hardware implementation). On the OpenMote-CC2538, instead, hardware accelerators leads to a gain of up to 58 % (for the hybrid implementation) and 66% (for the hardware implementation). Moreover, OpenMote-CC2538 always emerges as the more energy-consuming platform.

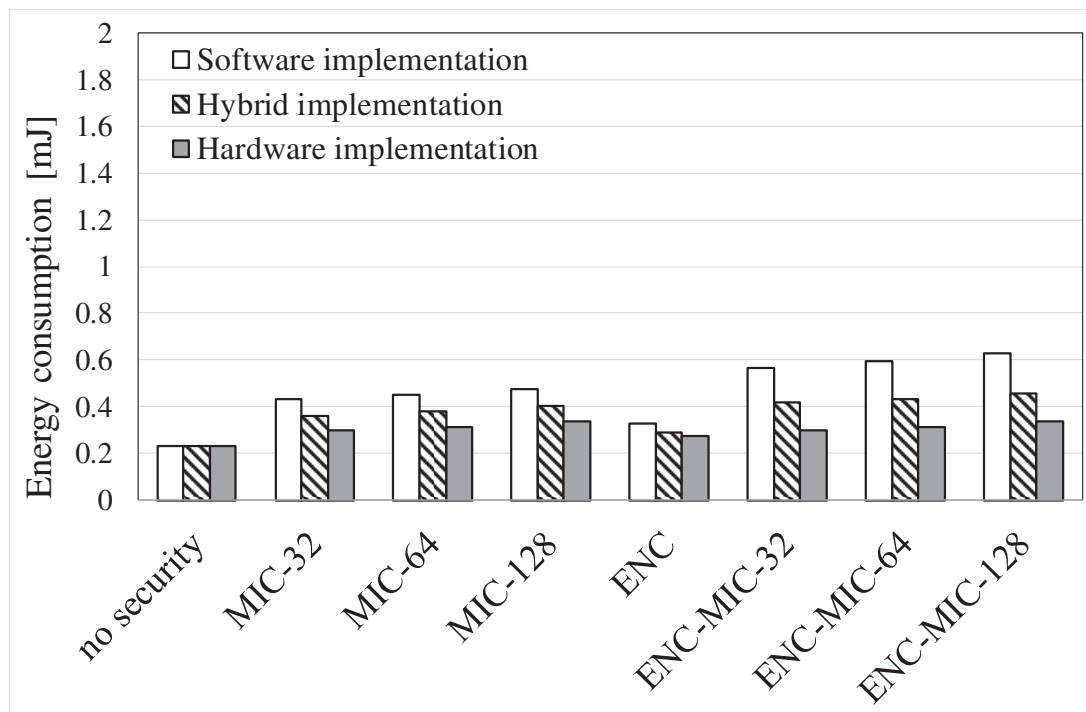


Figure 3.8: Energy consumed in a single timeslot for the TelosB mote.

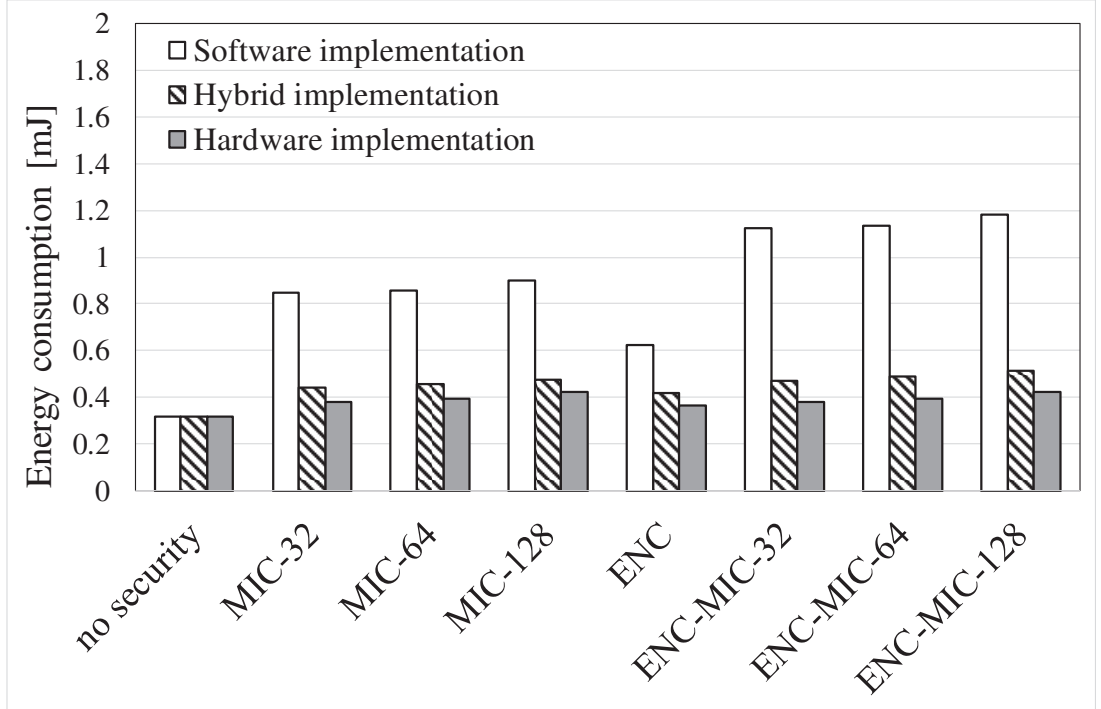


Figure 3.9: Energy consumed in a single timeslot for the OpenMote-CC2538 mote.

Note that this estimation is useful to demonstrate that the amount of energy consumed by motes is directly related with the way security is implemented and applied. Therefore, a live energy evaluation that takes care of real effects goes beyond the scope of this contribution and is left for future works.

3.2.6.5 Final considerations on high-level application design

To conclude, based on the results described in the previous Section, additional comments on high-level application design and energy consumption can be formulated.

As already anticipated by introducing this research activity, the shorter the timeslot, the higher the network throughput. Therefore, the provisioning of link-layer security inevitably brings to a throughput reduction. In addition, the message overhead produced by each security level (i.e., Auxiliary Security Header plus MIC, if present) differently impacts on the effective amount of bytes that can be inserted, in each packet, by the application layer. Practically, the data generation rate, R , that a high-level application

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

running on the i -th device could use should satisfy the following equation:

$$R \leq \frac{N_i A_j}{N_{ts} \text{TsSlotDuration}}, \quad (3.2)$$

where N_i , A_j , and N_{ts} are the number of timeslots assigned to the i -th device for transmitting data, the maximum size of an application message due to the j -th security level¹, and the total number of time-slots available in a TSCH slot frame (i.e., a predefined number of time-slots that repeats over the time). For simplicity, Eq. (3.2) has been obtained by assuming the absence of packets fragmentation at the application layer. Maximum performance are reached if $R = (N_i A_j)/(N_{ts} \text{TsSlotDuration})$ and no packet losses are registered on the wireless communication channel. Higher values of R are not supported because they produce an uncontrolled increment of packet queue, thus bringing to a growth of packet losses (e.g., packets are lost at the transmitter side because there is not enough space in the queue).

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

As shown in Section 3.2, the IEEE 802.15.4 standard allows to protect MAC packets by means of symmetric-key cryptography techniques and several security options. Nevertheless, it does not directly explain how to handle the initialization of a secure domain, the generation of layer-2 keys, as well as the orchestration of security services in a wide gamut of network configurations [50].

Of course, these important aspects have been already investigated in literature and standardization bodies, as detailed in Section 2.6.2. However, a deep experimental analysis (with cross comparisons), aimed at evaluating the impact that security features have on latencies and energy consumptions is mostly missing. In addition, no contribution explored so far the opportunities and subtleties that could arise from a joint integration of existing solutions, in a standard compatible way.

¹For instance, $A_{no\ security} = 568$ bits, $A_{MIC-32} = 520$ bits, $A_{MIC-64} = 488$ bits, $A_{MIC-128} = 424$ bits, $A_{ENC} = 424$ bits, $A_{ENC-MIC-32} = 520$ bits, $A_{ENC-MIC-64} = 488$ bits, and $A_{ENC-MIC-128} = 424$ bits

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

To complement the valuable efforts already put in place on layer-2 security in IEEE 802.15.4 networks, as well as to face the limitations described above, this section designs, implements, and experimentally evaluates (for the first time) an integrating framework, namely LICITUS, that harmonizes and orchestrates, the functionalities of consolidated approaches to *Security Configuration*, *Bootstrap*, and *Key Negotiation* phases. The resulting system offers: (i) a rich set of possible security configurations that can be enabled also in heterogeneous scenarios where protected communications may coexist with unsecured ones; (ii) an effective methodology for easing the initialization of secure IEEE 802.15.4 networks; (iii) a lightweight KMP for negotiating a layer-2 key between a node pair; (iv) resilience to several security attacks, and (v) a full compatibility with IEEE 802.15.4 technology [50].

LICITUS has been conceived as a distributed framework: starting from a set of cryptographic materials and configuration variables stored by the manufacturer or updated by the system administrator, each node is able to autonomously bootstrap security services and negotiate a link-level key with its neighbors without requiring the intervention of any remote and centralized server. Therefore, it natively promises good levels of scalability, also in high loaded scenarios.

As a preliminary analysis, the resilience of LICITUS against password guessing, replay, and MITM attacks has been proved through the ProVerif tool, i.e., a widely accepted automatic cryptographic protocol verifier¹. Then, to demonstrate its effectiveness, LICITUS has been implemented within the emerging OpenWSN protocol stack [44] and experimentally evaluated using the TelosB platform².

A comparison with respect to the reference ZigBee IP security architecture, which integrates the SKKE protocol for handling the key agreement mechanism [53], has been also provided. Note that the considered ZigBee IP security configuration has been chosen as a valid benchmark strategy for comparison because it shares many features with the proposed framework and, in particular, the ability to establish a secure network domain in which node pairs negotiate symmetric keys. Experimental results show that the introduction of security services generally incurs a not negligible computational effort, which worsens communication latencies and energy efficiency. However, they also demonstrate that LICITUS is able to overcome the SKKE protocol of the ZigBee

¹<http://prosecco.gforge.inria.fr/personal/bblanche/proverif/>

²http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

IP security architecture by speeding up the configuration of security services (up to 120%) and ensuring relevant energy savings (larger than 50%).

The work not only represents an important advancement to the state of the art, but it completely fits within the activity of the IETF 6tisch WG [76]. In fact, it remarkably extends preliminary works such as [101] and [89] by: (i) detailing components and functionalities of the devised security framework, (ii) evaluating its performance through an extended experimental campaign, and (iii) providing a cross comparison with another well-known solution.

In addition, the open source implementation of LICITUS is freely released¹ and shared with all researchers and practitioners working on layer-2 security-related aspects. Therefore, this work can trigger relevant implications in academia, industrial, and standardization fields.

3.3.1 Details on the proposed security framework

The Lightweight sCheme for IoT secUre communicationS (LICITUS) framework proposed hereby enables, configures, and manages layer-2 security services in a IEEE 802.15.4 network. All of these operations are executed in a distributed manner. Starting from a set of cryptographic materials and configuration variables stored by the manufacturer or updated by the system administrator, in fact, each node is able to autonomously bootstrap security services and negotiate link-level keys with its neighbors without requiring the control of any remote and trusted server. Therefore, specific procedures have been defined for computing keys for protecting broadcast messages (generally used in the bootstrap stage) and keys for protecting unicast packets. In addition, LICITUS covers all the possible security configurations available in both homogeneous networks (i.e., all devices support security features) and heterogeneous ones (i.e., protected communications may coexist with unsecured ones), as well as allows dynamic updates of network settings based on the security capabilities of network devices. To accomplish these challenging goals, each device belonging to the IEEE 802.15.4 network hosts an instance of the LICITUS framework, made of three main sub-systems, that are *Security Configuration*, *Bootstrap*, and *Key-negotiation managers* (see Figure 3.10). The *Security Configuration Manager* controls the level of security offered in the network, imposes minimum security requirements, and supports the execution of all the

¹The code is available from <http://telematics.poliba.it/security-iot>.

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

procedures handled by other components of the framework. It stores a set of security parameters and initial credentials, used for initializing the secure domain. The *Bootstrap Manager* executes the bootstrap procedure devoted to the initialization of the secured domain. Finally, the *Key-negotiation Manager* handles the KMP protocol and the algorithm that dynamically updates link keys during the time. In general, LICITUS can integrate different key negotiation schemes, like DH, ECDH, RSA, etc. [49][102]. However, to ease the description of procedures implemented by the *Key-negotiation Manager*, the well-known DH algorithm will be considered from this moment on.

The following subsections provide a thorough description of LICITUS components, their interactions, and resulting security services.

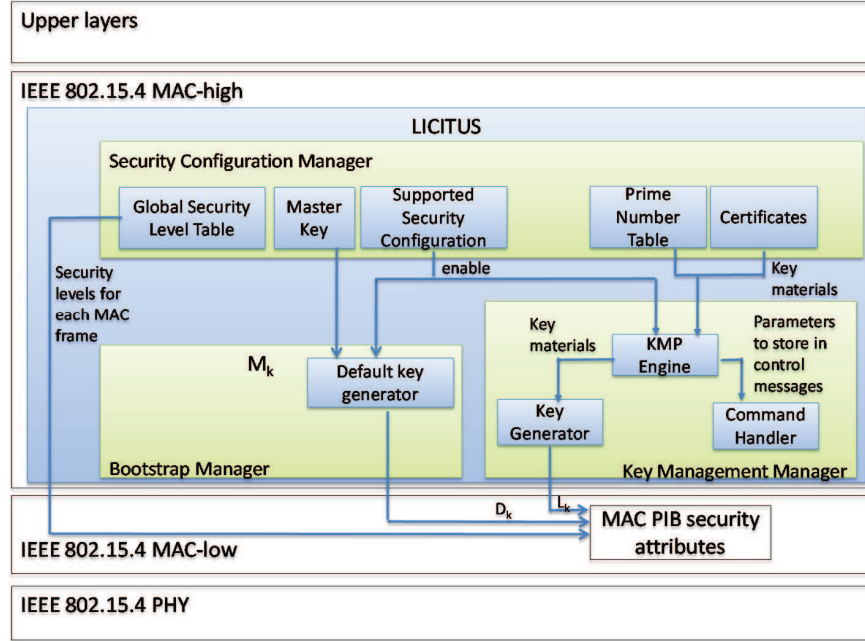


Figure 3.10: General scheme of LICITUS.

3.3.1.1 Security configurations

LICITUS natively supports all the security levels already defined in the IEEE 802.15.4 standard (and summarized in Sec. 2.6.1). For instance, it can be used for setting up a network where messages are exchanged in clear (level 0 security option), a network where communications are protected with the highest security level (supported by the

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

level 7 security option), or any other standard-compliant configuration. Also, according to the IEEE 802.15.4 standard, LICITUS allows multiple security options to be concurrently handled within the same network: a given data stream exchanged among a couple of devices may be potentially protected by a security level and a symmetric key that differ from those used for other kinds of broadcast/unicast communications [50].

From a security perspective, three different types of nodes can be found in IoT systems: (i) devices that *have* software/hardware components for handling cryptography operations and *are* in possess of initial credentials and parameters set needed to join a secured network, (ii) devices that *have* software/hardware components for handling cryptography operations and *are not* in possess of initial credentials and parameters set needed to join a secured network, and (iii) devices that *have not* software/hardware components for handling cryptography operations. To simplify, it is possible to assume that a node presents *security capabilities* if it has software/hardware components for handling cryptography operations and is in possess of initial credentials and parameters set needed to join the secured network. In this regard, LICITUS also supports heterogeneous scenarios, where nodes with and without security capabilities may coexist within the same network. Differently from homogeneous networks, where all exchanged frames can be protected, heterogeneous scenarios should support both *protected* and *unprotected* communications.

Now, to embrace any possible combination of these security options and configure homogeneous and heterogeneous scenarios, LICITUS introduces four high-level security configurations, that are:

- **Unsecured network:** security services are not enabled and all packets are exchanged in cleartext. This configuration can be natively used in heterogeneous scenario because the possession of security capabilities is irrelevant.
- **Fully Secured network:** all packets are both encrypted and authenticated. This configuration only supports homogeneous scenarios and nodes without security capabilities cannot join the network.
- **Partially Secured network:** only the message integrity is enabled. Also in this case, this configuration only supports homogeneous scenarios and nodes without security capabilities cannot join the network.

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

- **Hybrid Secured network:** secure and unsecure communications may coexist. This configuration is introduced for better supporting heterogeneous scenarios, hosting with and without security capabilities. Since from the beginning, the network is created in an unsecured manner. All the non-unicast control messages sent by the coordinator should be transmitted in clear, thus ensuring that all devices can read the packet contents. Moreover, a couple of nodes with security capabilities can establish a link-level secure communication and exchange unicasts and protected packets.

The relationship between high-level security configurations defined in LICITUS and security options standardized in IEEE 802.15.4 is reported in Tab. 3.4.

Table 3.4: Relationship between high-level security configurations and IEEE 802.15.4 security options.

High-Level Security Configuration	Security option used for broadcast messages (i.e., beacon)	Security option used for unicast messages
Unsecured	Level 0 only	Level 0 only
Fully Secured	From Level 5 to Level 7, according to system administrator choices	From Level 5 to Level 7, according to system administrator choices
Partially Secured	From Level 1 to Level 4, according to system administrator choices	From Level 1 to Level 4, according to system administrator choices
Hybrid Secured	Level 0 only	From Level 1 to Level 7, according to system administrator choices and device capabilities

To provide a further insight, the conceived framework also supports the possibility to dynamically change the security configuration; in the sequel, this will be referred to as *flexibility feature*. In particular, when the *flexibility feature* is enabled, a homogeneous and secured network, that falls in both *Partially Secured* and *Fully Secured* configurations, may move to the *Hybrid Secured* mode. In this way, it is possible to enable the join of external nodes not supporting security capabilities or not in possess of

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

the right initial credentials. In fact, when a node without security capabilities or initial credentials wish to join a network (that is already operating in a secured fashion), it has just to send a beacon request in clear. Its corresponding coordinator processes the request and switches to the *Hybrid Secured* configuration only if the *flexibility feature* is enabled. From that moment on, the considered node may join the network, all the broadcast messages are transmitted in clear, and nodes supporting security capabilities still continue to exchange protected unicast packets with their parents. Of course, this feature is optional and it can be enabled or disabled by the network administration according to the target design criteria.

Note that each of the aforementioned security configuration (including the *flexibility feature*) could be enabled by properly setting the *Security Level Table*, stored at the MAC level of the IEEE 802.15.4 standard [24], [21],[22].

3.3.1.2 Security Configuration Manager

As stated before, the *Security Configuration Manager* controls the level of security, imposes the corresponding minimum requirements, and supports all the procedures handled by other components of the framework. To this aim, the following parameters and initial credentials are stored:

- *Supported Security Configuration*: it represents the security configuration chosen for a given network among those in Sec. 3.3.1.1.
- *Master Key*: it is an initial secret shared among all the devices. Note that it is not directly used to encrypt/decrypt messages, but it is used, together with other time-varying parameters (that can be unique in each secured domain and periodically updated during the time) for generating all the required key materials. In particular, starting from the *Master Key*, two different keys are created: the *Default Key*, for protecting *broadcast* messages (i.e., beacon frames) and the *Link Key*, for encrypting and authenticating *unicast* packets (i.e., exchanged between a pair of specific nodes).
- *Global Security Levels Table*: it is a table that stores the minimum security level and the list of allowed security levels that should be adopted for each kind of MAC frame and security configuration. It should be created considering the type

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

of services offered by the IEEE 802.15.4 network. Section 3.3.1.5 presents some examples showing how to set up the Global Security Levels Table in practical use cases, that refer to industrial scenario and smart campus.

- *Prime Numbers Table*: it is a set of N prime numbers and their respective primitive roots used during the *Key Negotiation* phase for generating *Link Keys* according to the DH algorithm. Each available prime number is identified by $N_p = \log_2(N)$ number of bits.
- The *private key* of the device.
- A *Certificate*: it stores the public key of the device in a standardized structure (e.g., X.509 certificate [103]).
- *Public key of a certification authority*: it is used for verifying the authenticity of device certificates.

The *Security Configuration Manager* can be directly configured by the manufacturer or by the administrator before the deployment of the network. To ensure the protection against tampering attacks, specific software-based and/or hardware-based mechanisms may be used for preventing the physical access to all the stored variables [104].

3.3.1.3 Bootstrap Manager

The *Bootstrap Manager* is the entity of the LICITUS instance running on a given device that is in charge of initializing security parameters needed for joining a secure domain. Specifically, it processes the list of variables stored in the Security Configuration Manager and sets up security-related parameters at the MAC layer. In addition, it also calculates the *Default Key*, D_k , used for protecting broadcast messages and unicast messages exchanged for negotiating layer-2 keys. The *Default Key*, D_k , is computed starting from the *Master Key*, M_k , the coordinator MAC address, $MAC_{addr,c}$, and the network ID, PAN_{ID} , by using a 128-bit hash function, $H_{128}\{\cdot\}$:

$$D_k = H_{128}\{PAN_{ID}|MAC_{addr,c}|M_k\}. \quad (3.3)$$

The security level of the *Default Key* is strictly related to the specific hashing algorithm used to generate it. Without loss of generality, LICITUS may potentially adopt

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

any kind of hash function for generating the 128-bit digests. From one side, it is suggested to use robust hash functions, thus making the entire framework resilient against attacks. From another side, instead, it is even important to adopt hash functions that do not require high computational capabilities (that are generally scarce in constrained nodes forming a IEEE 802.15.4 network).

As a final consideration, it is worth remarking that the time instant when the *Default Key* is computed is different for network coordinator and child node. In particular, while the first device can generate the key as soon as it becomes the coordinator of a given portion of the LLN, the child node should firstly receive the beacon messages (i.e., association phase), to extract the parameters needed for the computation of the *Default Key*. Note that this task can be handled without any problem because these parameters are stored in the MAC header and, hence, transmitted in clear.

3.3.1.4 Key Negotiation Manager

The *Key Negotiation Manager* implements all the functionalities for the definition of the link key used for layer-2 unicast communications. It is made up of three entities: the *KMP Engine*, which implements the key negotiation scheme to derive a *Pre Link Key*; the *Command Handler* that translates logical messages of the KMP in MAC messages; and the *Key Generator* that generates all layer-2 keys, starting from the *Pre Link Key*.

As well known, very resource-constrained devices are unable to perform complex algorithms and protocols in a limited time, so that it is mandatory to implement simple and effective key agreement protocols [63]. For this reason, a lightweight approach with limited computational and bandwidth requirements has been developed.

The KMP is implemented in a distributed manner: a couple of communicating nodes can negotiate a layer-2 key without needing to interact with any remote and trusted node. The resulting approach is potentially scalable and it does not bring huge latencies in complex networks.

Now, without loss of generality, it is possible to consider two devices willing to negotiate a layer-2 key: node A and node B. For instance, A could be a child node directly connected to the network coordinator and B could be the network coordinator. The algorithm is initialized by node A and it is executed by exchanging four different logical messages. The first two messages deliver the key materials to be used by each

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

node involved into the key agreement scheme, for computing the *Pre Link Key* (i.e., a preliminary common secret which all the required link keys are generated from). The last two messages are used, instead, for finalizing the mutual authentication. Whereas the first two messages are protected (i.e., encrypted and/or authenticated) with the D_k , the last are always encrypted and/or authenticated by means of the *Pre Link Key* itself.

From the implementation point of view, these messages are stored within specific *Header Information Elements*, that are fields of the MAC header defined by the IEEE 802.15.4 standard for carrying customized information in a standard compliant way. During the execution of the KMP protocol, the *Command Handler* entity of the LICITUS framework translates high-level commands in specific link-level messages.

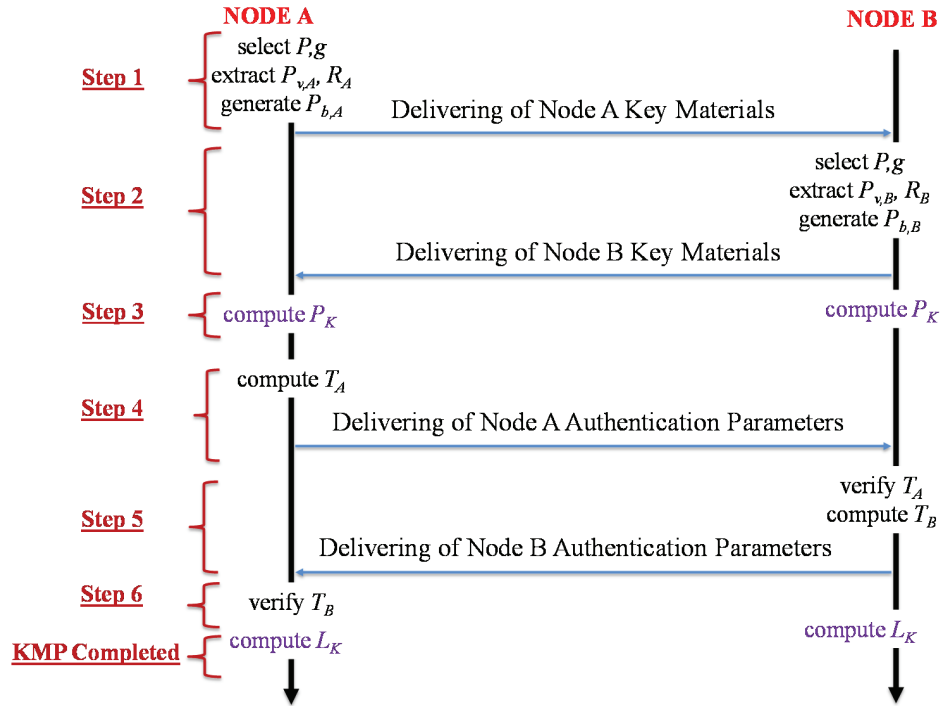


Figure 3.11: Message sequence chart of the designed KMP procedure.

The KMP supports both anonymous and certified DH schemes. It consists of six consecutive steps (see Figure 3.11).

- **Step 1:** The node A sends its public key, $P_{b,A}$, and a nonce, R_A , to the node B. While the former parameter is used for negotiating the key with the remote node,

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

the latter is used, instead, for handling the mutual authentication.

In the case of adoption of the anonymous DH, a prime number, P , and the corresponding primitive root, g , are firstly extracted from the *Prime Numbers Table* by considering the latest N_p bits from the output of the following hash function:

$$H_{128}\{PAN_{ID}|D_k\}. \quad (3.4)$$

Then, after having randomly generated the private key, $P_{v,A}$, the public key, $P_{b,A}$, is computed according to the DH algorithm:

$$P_{b,A} = g^{P_{v,A}} \cdot \text{mod} P. \quad (3.5)$$

When the certified DH approach is used, the public key is not computed from scratch; it is instead directly delivered by the certificate associated to the child node.

- **Step 2:** Also the node B sends its public key $P_{b,B}$ and a nonce, R_B , to the node A.

Similarly to the previous case, when the anonymous DH scheme is selected, a prime number, P , and the corresponding primitive root, g , are firstly extracted from the *Prime Numbers Table* (by following the aforementioned procedure). Then, the public key, $P_{b,B}$, is computed from the randomly generated private key, $P_{v,B}$, according to the DH algorithm:

$$P_{b,B} = g^{P_{v,B}} \cdot \text{mod} P. \quad (3.6)$$

Obviously, the public key is directly transmitted within a certificate if the certified DH approach is used.

- **Step 3:** The node A and the node B compute a *Pre Link Key*, P_k , using Eqs. (3.7) and (3.8), respectively.

$$P_k = P_{b,B}^{P_{v,A}} \cdot \text{mod} P, \quad (3.7)$$

$$P_k = P_{b,A}^{P_{v,B}} \cdot \text{mod} P. \quad (3.8)$$

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

- **Step 4:** The node A computes the authentication parameter, T_A , and sends it to the node B for the mutual authentication:

$$T_A = \begin{cases} H_{128}\{P_k||R_B||R_A\} & \text{for anonymous DH,} \\ E(P_k, S_{gn}) & \text{for certified DH,} \end{cases} \quad (3.9)$$

where

$$S_{gn} = S(P_{v,A}, H_{128}\{P_k||R_B||R_A\}) , \quad (3.10)$$

Note that $E(\cdot)$ and $S(\cdot)$ operators refer to the encryption and the digital sign algorithm, respectively.

- **Step 5:** The node B verifies the validity of the received T_A parameter. In affirmative case, it computes the authentication parameter, T_B , sending it to the node A:

$$T_B = \begin{cases} H_{128}\{P_k||R_A||R_B\} & \text{for anonymous DH,} \\ E(P_k, S_{gn}) & \text{for certified DH,} \end{cases} \quad (3.11)$$

where

$$S_{gn} = S(P_{v,B}, H_{128}\{P_k||R_A||R_B\}) , \quad (3.12)$$

in which the parameters R_A and R_B have been switched in the hash calculation, in order to demonstrate the possession of the private key.

- **Step 6:** The node A verifies the validity of the received T_B parameter.

Once all the steps have been executed, the child node and the coordinator are able to generate the *Link Key*, L_k .

As discussed in Section 3.2, the standard imposes to use the CCM* algorithm and a 128-bit key to protect MAC frames. At the same time, the CCM* algorithm assumes that each key must be used for a specific number of block ciphers (i.e., until the frame counter associated to a given communication reaches its maximum value). Hence, for the i -th group of block ciphers, the *Link Key*, L_k , is computed as:

$$L_k = H_{128}\{i||PAN_{ID}||P_k\}. \quad (3.13)$$

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

3.3.1.5 Details on the Global Security Levels Table

The configuration of the *Global Security Levels Table* is discussed below through practical examples. In particular, two reference use cases are taken into account: industrial scenario and smart campus.

In an industrial scenario, sensor nodes are spread into the environment for monitoring the temperature of rooms and machines, the movements of motors and components, the humidity, the light, and so on. The security of an industrial plant is generally critical. Data must be kept as more confidential as possible and all the communications should be authenticated. Also, the join of external devices is not admitted. Thus, the *Fully-Secured* network configuration is required. In this way, it is possible to ensure that (i) all packets exchanged in the network are encrypted and authenticated and (ii) nodes that do not support security capabilities cannot be part of the monitoring system.

A *Smart Campus* generally appears as a dynamic and heterogeneous scenario. The system may include fixed sensors deployed by the network administrator and external devices (think for example to mobile terminals of students, professors, technical workers, etc.) that temporally expose their resources. Fixed sensors may communicate in a secure way. They, in fact, are able to establish layer-2 secure links starting from the set of security parameters stored by the network administrator just before the network deployment. External devices could not be in possession of security capabilities. Therefore, they can just send and receive messages in clear. To handle the coexistence of these heterogeneous nodes, a *Smart Campus* should use the *Hybrid-Secured* network configuration. Broadcast packets (like the beacon message) should be transmitted in clear. Nodes with security capabilities can establish secured link-level communications. Other nodes, instead, can only exchange data in clear.

To conclude, technical details of parameters and specific values of the security levels stored in the *Global Security Levels Table* for both the considered use cases are reported in Tab. 3.5.

3.3.2 Performance evaluation and security analysis

A thorough evaluation of the LICITUS framework is reported in this subsection, along with extensive experimental comparisons with respect to the ZigBee IP security archi-

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

Table 3.5: Practical configurations of the Globals Security Levels Table.

Attribute	FrameType	Use case	
		Industrial scenario	Smart Campus
Security Minimum	Beacon	5	0
	Data	5	0
	Command	5	0
	Ack	5	0
Allowed Security Levels	Beacon	5,6,7	0
	Data	5,6,7	0,1,2,3,4,5,6,7
	Command	5,6,7	0,1,2,3,4,5,6,7
	Ack	5,6,7	0,1,2,3,4,5,6,7

tecture.

As a first step, a security analysis is presented for demonstrating the resilience of the proposed framework against different kinds of attacks. Then, the communication overhead incurred during the setup of a secure domain is evaluated. Finally, an experimental analysis is presented for assessing, in different topologies, the time and energy required to establish a secured domain.

3.3.2.1 Security Analysis

LICITUS, combined with the security capabilities already integrated within the IEEE 802.15.4 technology, is robust with respect to the most important and critical security issues, as cryptanalysis, tamper attacks, password guessing, replay attack, and MITM.

The security analysis discussed hereby starts from the cryptanalysis attack, which can be generally handled for obtaining layer-2 keys (i.e., the *Default Key* and the *Link Key*). The IEEE 802.15.4 technology uses AES-based cryptography algorithms. In general, AES is known as one of the most efficient and secured algorithm offering confidentiality services. Some attacks designed to force the AES scheme (see for example works presented in [105]) require long times and enormous computational resources, thus they are not effective in real contexts. Another category of attacks that can be successful in breaking the AES algorithm is the side-channel attack category. In this case, attackers use information gained from the physical implementation of the system

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

(e.g., timing information, power consumption, electromagnetic fields, etc.) in order to obtain the key. However, also in this case, the time required to perform such a complex attack is higher than the lifetime of link keys. As a consequence, LICITUS is intrinsically resilient to cryptanalysis issues.

Password-guessing attacks try to extract keys by means of dictionary-attacks [106]. LICITUS is extremely robust with respect to this kind of attacks because nodes do not use passwords for computing layer-2 keys. In fact, *Default* and *Link keys* are obtained from the *Master Key* and DH parameters, respectively.

A malicious user is able to obtain and/or negotiate link keys only accessing security parameters stored into the *Security Configuration Manager* component. In this context, it is very important to remark that tamper attacks are really important in IoT systems, especially when some nodes are physical accessible from an external attacker. As already anticipated in section 3.3.1.2, the proposed framework is integrable with specific software-based and/or hardware-based mechanisms (like those proposed in [104]) to prevent the physical access to all variables stored within the *Security Configuration Manager* component. However, details about mechanisms to prevent tamper attacks go beyond the scope of this PhD work.

The adoption of frame counters makes the IEEE 802.15.4 technology resilient against replay attacks. Since the proposed approach fully integrates IEEE 802.15.4 security features, it also inherits robustness to these threats.

Finally, regarding the MITM attack, it could be launched by either an internal device (i.e., a malicious node that knows secured secrets shared among the rest of network devices) or external nodes (i.e., that do not know the value of variables stored within the *Security Configuration Manager* component) for compromising the right execution of the key negotiation phase. The designed KMP uses well-known approaches already adopted in the past for other protocols, as Transport Layer Security (TLS). Hence, its security can be demonstrated by using existing analysis. In the first part of the protocol, the DH algorithm is used to negotiate the shared key. Here, MITM attacks can be avoided using X.509 certificates, used to uniquely bind the public key to its owner. The mutual authentication scheme implemented in the second part of the protocol, instead, protects the entire process against replay attacks. The aim of the two latest messages is inspired to functionalities provided by *Finished* messages in the TLS protocol [107]. Similarly to TLS, these packets carry an authentication field that

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

is computed by considering all values, included random numbers, exchanged with the first couple of messages. Thus, the security proof related to the proposed protocol is as for the TLS protocol [108][109].

Replay and MITM attacks are generally performed to compromise the KMP protocol and the mutual authentication between communicating nodes. To further validate the resilience of the proposed framework against these issues, the effectiveness of LICITUS has been tested through a widely accepted automatic cryptographic protocol verifier developed at Inria: the ProVerif tool [110].

Several scientific contributions, like [111, 112, 113, 114], already used ProVerif for analyzing the security of their proposals. With ProVerif, it is possible to generate a formal model of a security protocol leveraging different cryptographic primitives, including shared and public key cryptography, hash functions, and Diffie-Hellman key agreements [115]. Security analysis, instead, is done by using the Dolev-Yao model, i.e., a baseline procedure reproducing many operations that could be done during an attack (i.e., capture and modification of the stream of messages exchanged between two devices over an unsecured channel). Starting from a set of initial assumptions (for instance, the attacker only knows the algorithm, the attacker also knows the master key, and so on), the tool verifies if a malicious node is able to successfully force the KMP procedure, thus compromising the mutual authentication property, or not.

With reference to the KMP depicted in Fig. 3.11, the formal model developed for the ProVerif tool¹, contains the following main functions:

- *begin_Node_A ()*, meaning that the KMP procedure is initiated by node A;
- *end_Node_A ()*, meaning that the KMP procedure is completed by node A with node B;
- *begin_Node_B ()*, meaning that the KMP procedure is initiated by node B;
- *end_Node_B ()*, meaning that the KMP procedure is completed by node B with node A;

Moreover, some important outputs provided by ProVerif appear as reported below:

¹To allow the reader to verify the validity of the conducted test, the code is freely available at <http://telematics.poliba.it/security-iot>

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

- *not_attacker(MasterKey[]) is true.* It means that the malicious node is not in possess of the master key;
- *not_attacker(MasterKey[]) is false.* It means that the malicious node is in possess of the master key;
- *inj-event(last event name ()) ==> inj-event(previous event name ()) is true.* It means that a given operation, i.e., *last event name*, ends after that another operation, i.e., *previous event name*, is really executed. The operations *last event name* and *previous event name* can refer to one of the aforelisted functions.
- *inj-event(last event name ()) ==> inj-event(previous event name ()) is false.* It means that as soon as a given operation, i.e., *last event name*, is completed, it is not possible to ensure that another operation, i.e., *previous event name* has been really executed in the past. This condition identifies a security issue and envisages the possibility to have a successful attack.

The security analysis of the conceived KMP has been conducted by means of three different tests.

The first test assumes that the key agreement protocol is based on the anonymous DH algorithm and the attacker does not know the *Master key*. As reported in Fig. 3.12(a), ProVerif demonstrates that a malicious node is not able to compromise the right execution of the KMP. Thus, the mutual authentication is always guaranteed. In fact, it is registered that: node B completes the KMP when the procedure is really initiated by node A, and node A completes the KMP when the procedure is really initiated by node B.

The second test assumes that the key agreement protocol is based on the anonymous DH algorithm and the attacker knows the *Master key*. Differently from the previous case, the malicious node can successfully compromise the right execution of the KMP. In fact, as reported in Fig. 3.12(b), ProVerif realizes that when node B completes the KMP, it is not possible to ensure that the procedure was really initiated by node A. Indeed, it could be initiated by the malicious node. Similar considerations can be argued in the case A is the node that completes the KMP.

The third test assumes that the key agreement protocol is based on the certified DH algorithm and the attacker knows the *Master key*. As reported in Fig. 3.12(c), ProVerif

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

demonstrates that a malicious node is not able to compromise the right execution of the KMP. Therefore, the mutual authentication is still guaranteed. In fact, it is registered that: node B completes the KMP when the procedure is really initiated by node A, and node A completes the KMP when the procedure is really initiated by node B.

To sum up, when the master key is secret, LICITUS is resilient against any attacks aiming at compromising the mutual authentication (like man-in-the-middle, replay, etc.). Furthermore, when the secrecy of the master key is compromised, the mutual authentication is still guaranteed only in the case the KMP makes use of X.509 certificates.

```
sciancalepore@telematics-Inspiron-3542: ~/proverif1.93
sciancalepore@telematics-Inspiron-3542:~/proverif1.93$ ./proverif docs/licitus_kmp_dh.pv | grep "RESULT"
RESULT inj-event(end_Node_B(pk,x_69)) ==> inj-event(begin_Node_A(pk,x_69)) is true.
RESULT inj-event(end_Node_A(pk_1739,x_1740)) ==> inj-event(begin_Node_B(pk_1739,x_1740)) is true.
RESULT not attacker(MasterKey[]) is true.
```

(a)

```
sciancalepore@telematics-Inspiron-3542: ~/proverif1.93
sciancalepore@telematics-Inspiron-3542:~/proverif1.93$ ./proverif docs/licitus_kmp_dh.pv | grep "RESULT"
RESULT inj-event(end_Node_B(pk,x_69)) ==> inj-event(begin_Node_A(pk,x_69)) is false.
RESULT (even event(end_Node_B(pk_3218,x_3219)) ==> event(begin_Node_A(pk_3218,x_3219)) is false.)
RESULT inj-event(end_Node_A(pk_3438,x_3439)) ==> inj-event(begin_Node_B(pk_3438,x_3439)) is false.
RESULT (even event(end_Node_A(pk_7221,x_7222)) ==> event(begin_Node_B(pk_7221,x_7222)) is false.)
RESULT not attacker(MasterKey[]) is false.
```

(b)

```
sciancalepore@telematics-Inspiron-3542: ~/proverif1.93
sciancalepore@telematics-Inspiron-3542:~/proverif1.93$ ./proverif docs/licitus_kmp_dh_cert.pv | grep "RESULT"
RESULT inj-event(end_Node_B(pk,x_69)) ==> inj-event(begin_Node_A(pk,x_69)) is true.
RESULT inj-event(end_Node_A(pk_4614,x_4615)) ==> inj-event(begin_Node_B(pk_4614,x_4615)) is true.
RESULT not attacker(MasterKey[]) is false.
```

(c)

Figure 3.12: Screen-shots of the output provided by the ProVerif automatic tool a) with the anonymous DH and the Master Key secret b) with the anonymous DH and the Master Key widely known and c) with the certified DH and the Master Key widely known.

3.3.2.2 Preliminary investigation

The number of logical and MAC messages that a pair of devices have to exchange for establishing a secure communication represents the first term of comparison between the LICITUS framework and the ZigBee IP security architecture.

As depicted in Figure 3.11, LICITUS always requires 4 logical messages. Considering that IEEE 802.15.4 specifications impose a MTU equal to 127 bytes, these messages

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

are mapped into 4 MAC packets when the KMP is based on the anonymous DH scheme. Instead, when the certified DH approach is adopted, the number of MAC packets becomes 24. In fact, by storing the public key in a X.509 certificate [103] of 864 bytes¹, the first two logical messages defined in the KMP scheme need to be fragmented in multiple MAC packets.

Regarding the ZigBee IP specifications, recall from Section 2.5.3 that the key agreement is handled by means of the SKKE protocol [53]. It uses a centralized scheme where the *Trust Center* always coordinates the negotiation of keys among any couple of nodes, resulting in a total number of 9 messages necessary to establish a shared key between any couple of devices.

Supposing to use the anonymous DH scheme as the key negotiation algorithm, it can be immediately noted that the number of MAC packets required to establish a secured communication in ZigBee IP (i.e., 9) is higher than those needed by the proposed solution (i.e., 4). In addition, recall that in Zigbee IP some messages (i.e., the second, the third, and fourth in Figure 2.1) are exchanged with the *Trust Center* that may be some hops away from the nodes involved in the KMP. As it will be demonstrated in the following section, this will negatively impact on energy consumptions and latencies. It is worth to note that, the designed KMP protocol requires an higher number of MAC messages when the certified DH scheme is used (that are equal to 24 packets as discussed before). This is the cost to pay for offering also the authentication of devices; but, this is an important feature not supported by SKKE.

Anyway, apart these preliminary comments, the time and the amount of energy needed to configure a secure system are strictly influenced by the network load, packet losses, de-synchronization issues, and complexity of each task, as discussed in the rest of the subsection.

The reason behind the high number of messages required by the SKKE protocol can be found in the nature of devices that is supposed to perform the protocol. In fact, the SKKE protocol is assumed to work on devices that have minimal computational and storage capabilities. Therefore, the security can be constructed only by performing many interactions with the only node that can have more processing capabilities than

¹it is supposed to create a X.509 certificate by using the Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

the others, that is the Trust Center. As a consequence, the protocol increases the number of messages while lowering the processing capabilities required on the nodes.

3.3.2.3 Experimental settings

To carry out experimental tests, LICITUS and the security mechanisms defined in ZigBee IP specifications have been implemented in the OpenWSN solution. In particular, components and functionalities of the proposed framework have been developed on top of the security extension presented in [89][25].

Details about ROM and RAM footprint of the considered security solutions have been summarized in Table 3.6. From reported values it is possible to observe that the proposed approach guarantees the minimum memory demands, thus becoming very suitable for any kind of constrained device.

Table 3.6: Memory requirements.

Method	ROM usage [B]	RAM usage [B]
Zigbee IP	5390	18
LICITUS	5122	6

Experimental testbeds have been set up using TelosB devices, equipped with a 16-bit microcontroller, working at a maximum speed of 8 MHz, 48 kByte Flash Memory, 10 kByte RAM, and a CC2420 radio module. These devices, being very constrained, allow to test LICITUS under very strenuous conditions.

Regarding the hash function, the tests use an open-source implementation of the SuperFastHash function¹. By considering the computational capabilities of the TelosB mote, it ensures a good compromise among simplicity, footprint, security and computational overhead.

Three different network topologies have been considered:

- **Star Topology**, composed by one coordinator and up to 10 child devices directly connected to it (i.e., there are only one-hop connections).
- **Chain Topology**, composed by up to 17 devices connected by means of a multi-hop path.

¹<http://www.azillionmonkeys.com/qed/hash.html>

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

- **Binary Tree Topology**, composed by 7, 15, and 31 devices. In that case, the maximum number of hops from the leaf node and the coordinator is equal to 2, 3, and 4, respectively.

In order to really realize multi-hop connections in both chain and binary tree topologies, devices have been placed sufficiently far, in order to ensure that nodes not directly connected at the layer-2 do not interfere with each other.

Focusing the attention on the MAC layer, and in line with [33], the IEEE 802.15.4 network has been configured to guarantee a slot lasting 15 ms and a superframe made of 101 slots. Moreover, in each superframe, the first slot is used to exchange advertisement packets; the next five slots are configured in shared mode for sending and receiving keep-alive packets and routing messages, and only one slot is shared between each couple of devices for data transmissions. Instead, at the physical layer the power transmission level has been set to -25 dBm, which brings to a maximum transmission range equal to around 30 cm.

Regarding the LICITUS framework, the network has been set in the *Fully Secured* configuration, adopting a KMP protocol based on the anonymous DH scheme (as discussed above, this ensures same operational conditions with respect to the ZigBee IP architecture). For the ZigBee IP security architecture, instead, all the functionalities of the *Trust Center* have been integrated in the network coordinator.

Particular attention has also been dedicated to the realization of the energy measurement system, used for estimating the amount of the current drained by TelosB batteries during the time. In detail, it is composed by:

- a desktop computer equipped with the LabVIEW tool, i.e., a software providing an easy and programmable interface between the computer itself and remote measurement equipments.
- A data acquisition card (model 12-bit NI DAQCard-6024E) that enables the interaction between the aforementioned computer and a shielded connector block described below. It is in charge of delivering data sampling instructions from the computer to the probe and data in the reverse path.
- A shielded connector block (model BNC-2120) with signal-labeled BNC connectors, that interface the data acquisition card to the reference resistor through the BNC-to-clip adapter.

3.3 LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

- A $1\ \Omega$ reference resistor, used to effectively measure the drained current.
- A probe that creates a physical connection between the reference resistor and TelosB batteries.

Starting from the current measurements, the amount of energy consumed by a single device is computed as:

$$E = \sum_i I_i V \Delta T, \quad (3.14)$$

where I_i , V , and ΔT are the i -th sample of the measured current, the voltage of the batteries (i.e., 3 V), and the sampling time interval (i.e., $50\ \mu\text{s}$), respectively.

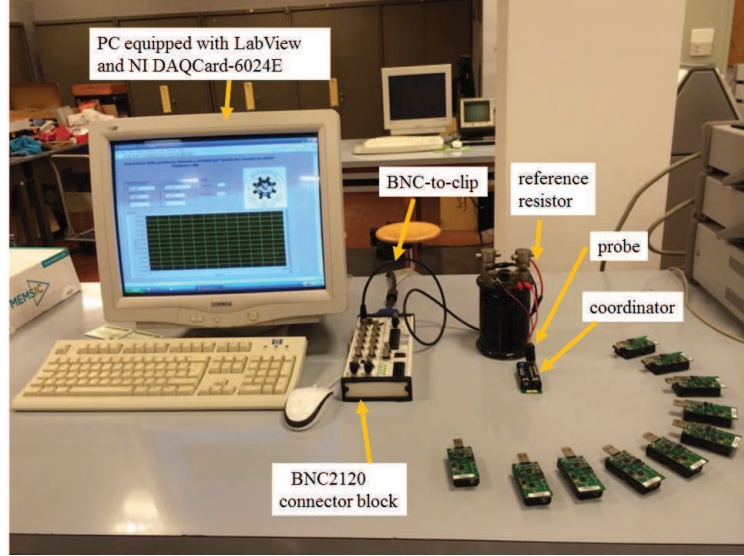


Figure 3.13: Energy measurement system.

Finally, all the tests have been run 100 different times. Moreover, it has been verified that the confidence intervals computed from obtained results are limited. This means that conducted tests are enough for ensuring the goodness of conducted experimental evaluations.

3.3.2.4 Time and energy needed for configuring security services

Time and energy consumptions incurred during the configuration of security services have been evaluated in different topologies by varying the number of nodes. All nodes have been configured for starting the key agreement scheme with their parent as soon

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

as they complete the association phase. In the ZigBee IP security architecture, the SKKE protocol starts immediately after the reception of the beacon message. In the LICITUS security framework, instead, the KMP is executed only after the generation of the *Default key* by the *Bootstrap Manager*.

The time needed to establish a secured network is reported in Figure 3.14.

As a general comment, it can be observed that the configuration of security services always brings not negligible computational efforts. In fact, it emerges that the higher the network size, the higher the amount of time required to initialize security services and to negotiate layer-2 keys. However, obtained results clearly demonstrate that LICITUS always ensures the lower airtime consumption, thus emerging as a promising, efficient, and scalable solution.

SKKE registers worse performance for two reasons. First, the key negotiation scheme requires the exchange of an higher number of messages. Second, the entire procedure is coordinated by a central entity, the Trust Center, that introduces an additional latency. Furthermore, it is evident that SKKE presents serious scalability issues: the amount of time required to configure security services drastically grows when the network size increases (see, for instance, the chain of 16 hop counts and the binary tree with 10 hop counts). As a result, SKKE does not scale with the number of nodes in the network.

On the contrary, by using distributed and lightweight procedures for network bootstrapping and key negotiation, LICITUS potentially ensures good level of scalability. However, also for the proposed approach, it is possible to observe that the amount of time needed to set up a secure network increases with the number of nodes. Different reasons can be argued for different network topologies. In a star topology, the network coordinator should establish layer-2 keys with many nodes. Unfortunately, constrained nodes cannot execute multiple instances of the KMP at the same time, due to their limited computational and memory capabilities. Accordingly, different KMP processes are serialized and the resulting airtime consumption increases as well. By generalizing, isolated scalability issues can be found when many constrained devices would like to establish, at the same time, a layer-2 key with the same parent node. Of course, such a problem can be fixed when the network hosts more capable devices, able to run several KMP instances in parallel. In network topologies involving multi-hop connections (including both chain and binary tree), the airtime consumption increases because the

time needed to set up the entire architecture at both MAC and network layers. For instance, in a chain with 10 hop counts, node at hop 10 can establish a secure communication link with node at hop 9 only after that node at hop 9 completed the same procedure with respect to node at hop 8.

The analysis of the amount of energy consumed by each node of the network, reported in Figure 3.15, fully confirms all of the comments reported above. In this case, the energy spent by the network coordinator, the leaf child, and the intermediate node (namely *parent* in Figure 3.15) have been measured. The coordinator, which is in charge of handling the most of tasks in the network, always incurs the highest energy consumption. On the contrary, child devices experience the lowest energy consumptions due to the lower number of operations they manage during the time. However, slightly larger energy consumptions are registered for intermediate nodes that, differently from the leaf child, have to manage key negotiation mechanisms with their parent and child nodes.

Furthermore, the reported investigation clearly demonstrates, once again, that LICITUS is able to configure security services in a consistently less time than the ZigBee IP security architecture, while guaranteeing more than the 50% of energy savings.

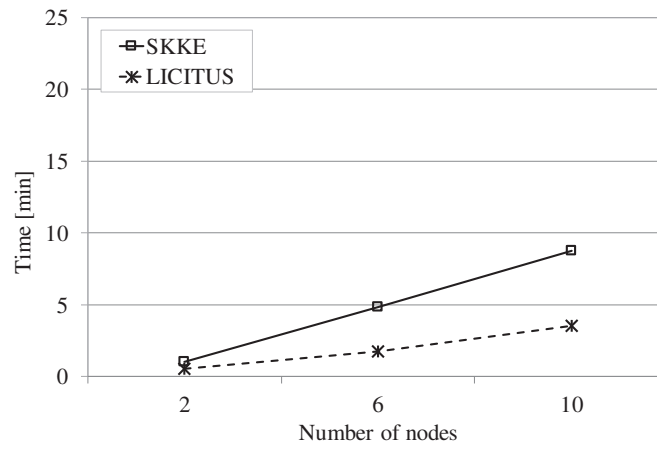
To provide a further insight, the impact that such a energy consumption has on a network operating in the long run has been investigated. The study, however, demonstrated that the amount of energy consumed during the initialization of security services when both LICITUS and SKKE are used has a limited impact on the network lifetime. In fact, it is always less than 1% of the whole battery capacity. Nevertheless, in the case the application scenario requires that security services must be periodically renewed, the energy saving reached by LICITUS may further amplify its advantages, thus increasing the network lifetime. Anyway, being out of scope of this contribution, it is possible to leave a deep analysis of this specific aspect as a future work.

3.4 A real-world application: TLSensing

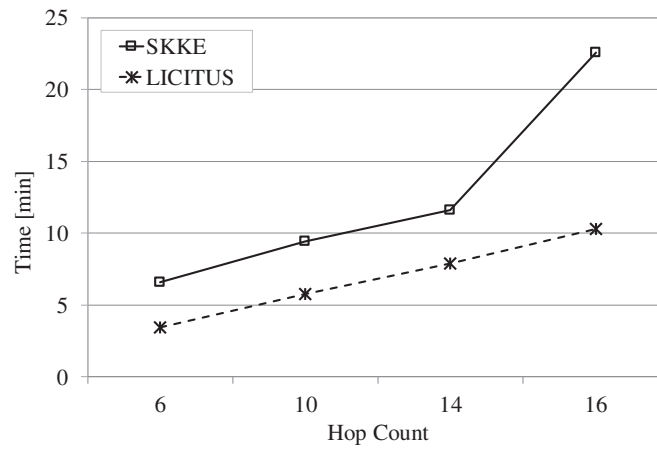
The work done in the layer-2 security scope, described in detail throughout the sections 3.1, 3.2 and 3.3, can be integrated within any real-word application based on the 6tisch protocol suite, providing many enhanced functionalities:

- procedures and mechanisms to secure and un-secure layer-2 frames;

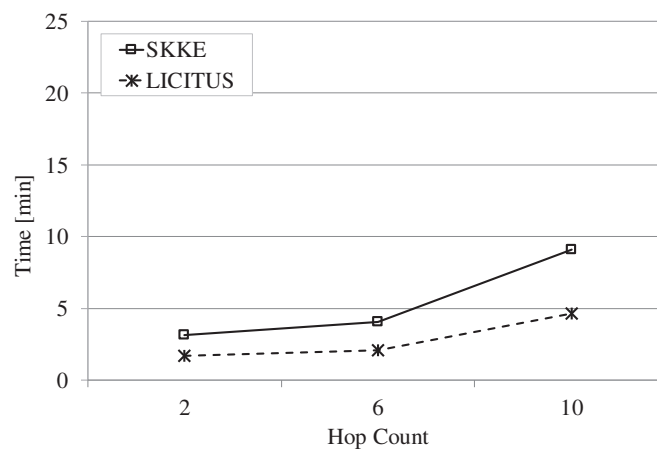
3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS



(a)



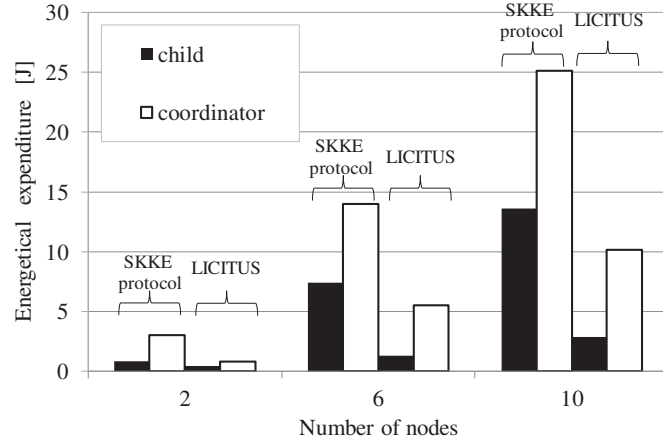
(b)



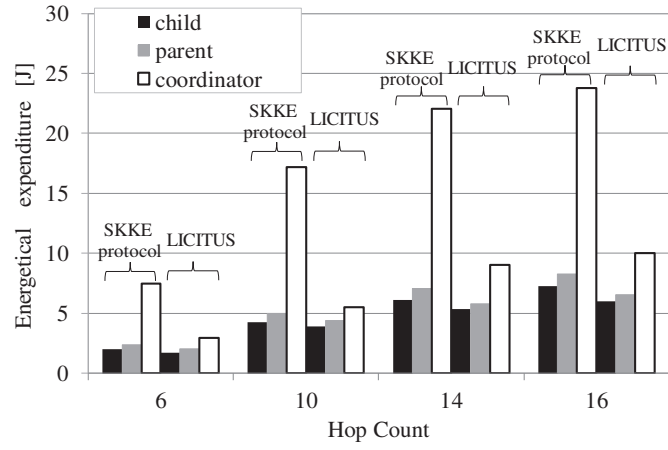
(c)

Figure 3.14: Time required for configuring security services in: (a) star, (b) chain, and (c) binary tree topologies.

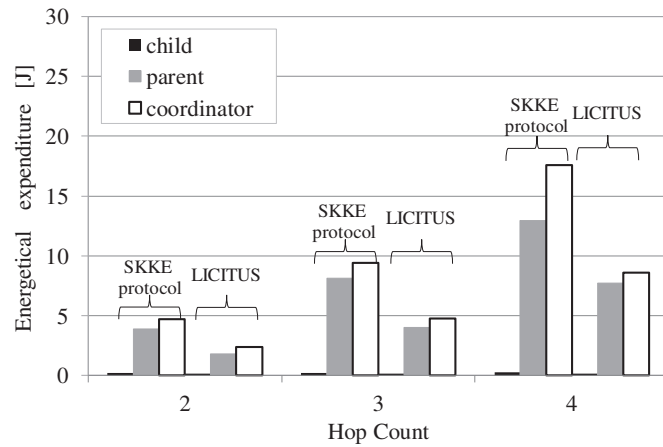
3.4 A real-world application: TLSensing



(a)



(b)



(c)

Figure 3.15: Energy consumptions for configuring security services in: (a) star, (b) chain, and (c) binary tree topologies.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

- authenticated and encrypted wireless communications between any couple of neighboring devices;
- mechanisms for handling the initialization of a secure IEEE 802.15.4 domain;
- a lightweight KMP for deriving layer-2 keys for every couple of neighboring devices.

A practical example of a proof-of-concept testbed integrating these functionalities is *TLSensing*, an open-source system that provides real-time data acquired from constrained devices and expose them through a friendly web-interface. It has been described in [116] with reference to a possible application as an IoT-based measurement system for aerial vehicles.

The *TLSensing* system is composed by a number of constrained devices equipped with sensing and wireless communications capabilities, a central system coordinator that controls data acquisition processes through a dedicated monitoring software, and a web server application that collects data coming from the network and makes them available to the user. The communication among the central system coordinator and constrained devices is enabled by low-power and short-range wireless technologies based on IEEE 802.15.4 and IEEE 802.15.4e standards. At the application layer, instead, the interaction between involved entities is handled through a client-server paradigm, built on top of open standards emerging from IETF standardization bodies, as described in Section 1.3.5. The web server, that runs on the same machine hosting the monitoring software application, exposes collected data to users and external softwares belonging to the health management system by using a web-based/user-friendly interface and a dump of the database structured according to the JavaScript Object Notation (JSON) syntax. Indeed, after having carefully described the main features offered by the developed *TLSensing* platform, a presentation of how *TLSensing* works is showed by some measurements collected with an experimental testbed deployed in the *TelematicsLAB*¹ research laboratory.

¹<http://telematics.poliba.it/>

3.4.1 TLSensing in a nutshell

The core of *TLSensing* is a wireless sensor network made up of a number of constrained devices, having both sensing and communication capabilities, deployed around the aircraft. They are in charge of gathering data from the environment (like temperature, humidity, luminosity, and acceleration) and make them available to the rest of the monitoring platform. In line with the protocol stack described in the Section 1.3.5, the communication technology is based on IEEE 802.15.4 and IEEE 802.15.4e standards and the adopted protocol stack. Moreover, in each constrained device is installed the OpenWSN stack [44]. Recall from the description in Section 1.4.2.1 that OpenWSN is an open source implementation of the protocol suite reported in Fig. 1.8. In order to guarantee confidentiality and authenticity services, messages are exchanged at the layer-2 in a secure fashion. In particular, the layer-2 payload is encrypted and authenticated by using security functionalities already defined in the IEEE 802.15.4 and IEEE 802.15.4e specifications; the whole network works in the *Fully Secured* configuration [50], and link keys are negotiated among nodes by using the lightweight key management protocol proposed in [50] and described in detail in Section 3.3. In general, *TLSensing* does not impose any constraint to the network topology and extension, thus fully supporting (just to make an example) star, chain, binary, and mesh configurations. The whole wireless sensor network is coordinated by a central node, the *PAN coordinator*, which is connected to the central system coordinator through an USB interface. Indeed, it acts as a border router between the IoT domain and the external Local Area Network (LAN). The central system coordinator runs a monitoring software that controls data acquisition processes by adopting a client-server communication paradigm. The software is the client that periodically generates requests to send towards remote constrained devices. On each constrained device, instead, runs an application server that processes these requests, queries on-board sensors, and delivers measurement data back to the central system coordinator. As it will be better explained in the sequel, messages are encapsulated into the CoAP protocol and the *PAN Coordinator* just acts as a relay between the client and the server. Data retrieved by the monitoring software are finally delivered to the web server application that will make them available through a web-based/user-friendly interface. This means that the user may visualize the outputs of the measurement process on a web-page by using any device connected through the

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

LAN. The history of captured data is continuously stored within the database, whose dump is exposed through a specific JSON-based data structure. This makes easier the integration of *TLsensing* in complex and general health management systems.

To summarize, the *TLsensing* network architecture has been shown in Fig. 3.16.

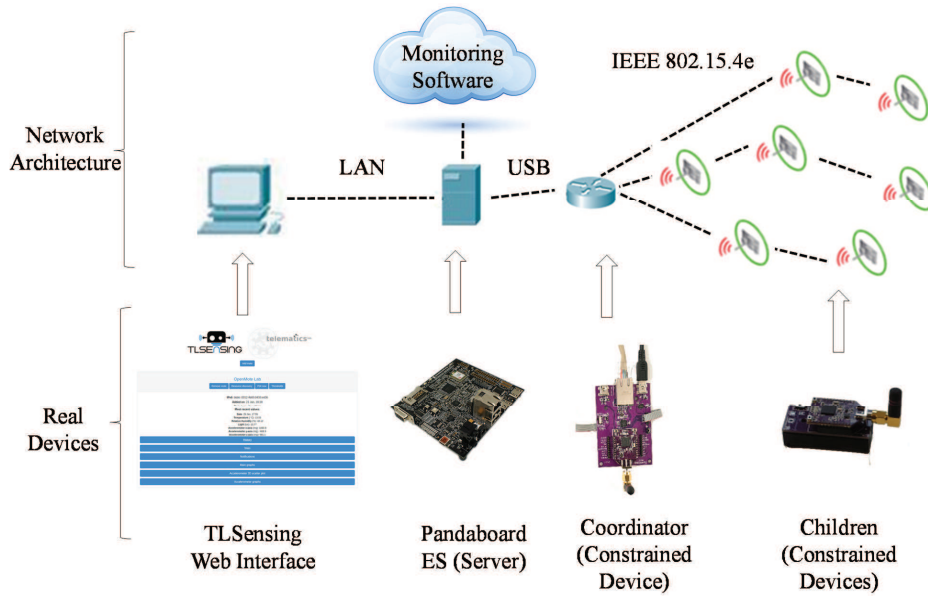


Figure 3.16: Network architecture.

The web server application also allows the user to interact with the measurement system (i.e., by observing the set of measured data or forcing an update of information acquired in the past). Once the network is deployed, the user may discover the list of constrained devices that are available in the aircraft by clicking on the *resource discovery button* available on the web page of the application. From this moment on, data can be retrieved by means of two mechanisms: from one side, the monitoring software polls remote devices every 10 s, thus triggering them to provide an answer with a message containing the current value of variables measured by their sensors; from another hand, the user can use the *poll now button* to force the poll of remote nodes, thus capturing any measured parameter at a desired time instant. Received measurements are then processed and visualized on the web page of the application. In summary, the following sections can be seen from the web page (see for example Fig. 3.17):

3.4 A real-world application: TLSensing



Figure 3.17: TLSensing web page.

- history section, which stores the latest values polled from a specific device (as default it reports the latest 10 values);
- statistics section, which reports the maximum, the minimum, and the mean values of aforementioned values;
- notification section, which reports possible error messages (e.g., the retrieved data exceeds a pre-defined threshold);
- graphs section, which graphically shows the trend of retrieved measurements;
- accelerometer 3D scatter plot, which draws the scatter plot related to a given remote device;
- accelerometer graphs, which details the values provided by the accelerometer, both in x, y and z directions.

Details on hardware, firmware and software components used to build up the *TLSensing* architecture will be described in what follows. Further-

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

more, softwares, firmwares, and deploying instructions are freely available from http://telematics.poliba.it/tlsensing_tool.

3.4.1.1 Hardware details

The central system coordinator has been installed on the Pandaboard ES platform, which is a low-power and low-cost single-board computer development platform. It is an open-source development platform which integrates dual-core 1.2 GHz ARM Cortex-A9 MPCore CPU, 384 MHz PowerVr SGX540 GPU, IVA3 multimedia hardware accelerator with a programmable DSP, and 1GiByte of DDR2 SDRAM, as well as SD Card slot, 10/100 Ethernet, Wi-Fi, and Bluetooth interfaces, output video signal via DVI and HDMI interfaces, and two USB ports.

Constrained devices are, instead, provided by the OpenMote technologies¹ described in Section 1.4.1.3.

3.4.1.2 Firmware and software details

All the functionalities of the proposed *TL Sensing* are enabled thanks to a set of applications implemented in both constrained devices and central system coordinator.

From one hand, a CoAP server application, namely *SensorData*, has been developed in the firmware of the constrained devices. In particular, it has been implemented starting from the OpenWSN protocol stack (version 1.9.0 has been considered as a reference). *SensorData* starts in the initialization phase of the protocol stack and continuously waits for GET requests directed to the */sens* path. Moreover, GET requests may also contain a specific set of arguments, through which generating customized queries (i.e, poll all sensors vs poll a single sensor directly).

From another side, the monitoring software integrates an open-source library, namely *OpenCoAP*, that is used to interact with remote devices by means of CoAP messages. The possibility to generate GET requests with customized queries is enabled by the URI-Query options, already integrated within the aforementioned library. Finally, the web server application is written in Python, is built on top of the well-known Django framework², and it makes use of the non relational MongoDB database³.

¹<http://www.openmote.com/>

²<https://www.djangoproject.com/>

³<http://www.mongodb.org/>

3.4.2 TLSensing at work

With the aim of describing how does *TLSensing* work, an example showing the interaction between user and constrained device has been reported in Fig. 3.18. From the

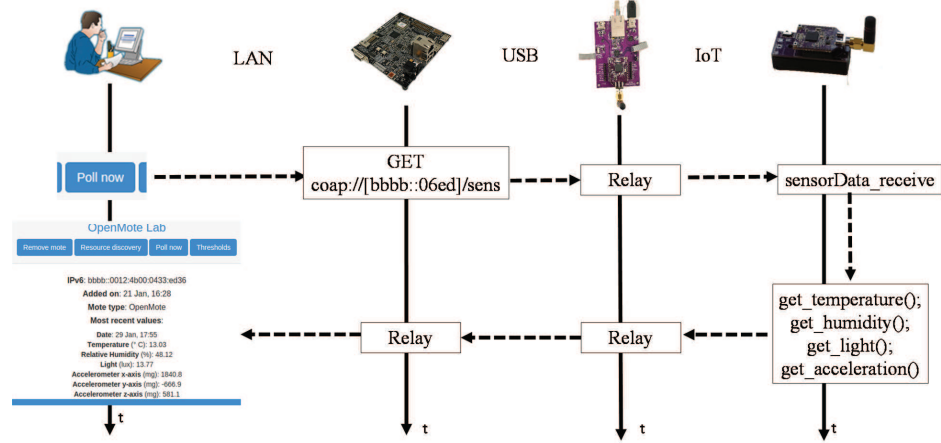


Figure 3.18: Interaction between the user and the measurement system.

illustration, it is possible to observe that the user generates requests by clicking on the *Poll Now* button. The web server application contacts the monitoring software that triggers the generation of a CoAP GET request to send towards a specific constrained device. Such a request is delivered via the *PAN Coordinator* to the destination node, that takes required data from on-board sensors and generates a CoAP response to send back to the client (i.e., the monitoring software). Once the requested data are received, they are then processed and visualized on the web page.

In addition, to demonstrate how *TLSensing* works in a practical scenario, an experimental testbed has been deployed in the TelematicsLAB research laboratory, thus allowing any user to see, in real-time, local measurements. The testbed is available from <http://telematics.poliba.it/tlsensing>.

For example, Figs. 3.19 and 3.20 summarize temperature and relative humidity measurements gathered during a single day by motes located indoor and outdoor, respectively. As expected, values provided by the indoor mote do not significantly change during the time. The mote placed outside, instead, reports variable values that strictly depend on external conditions.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

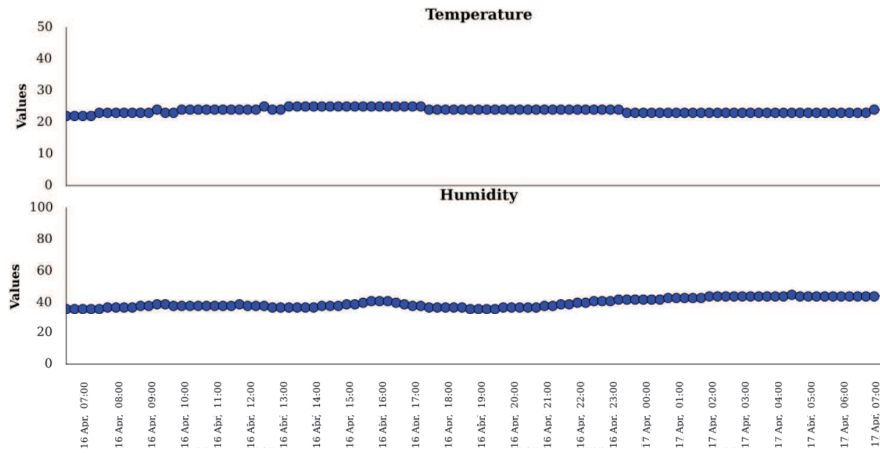


Figure 3.19: Temperature and relative humidity acquired in an indoor environment during a single day.

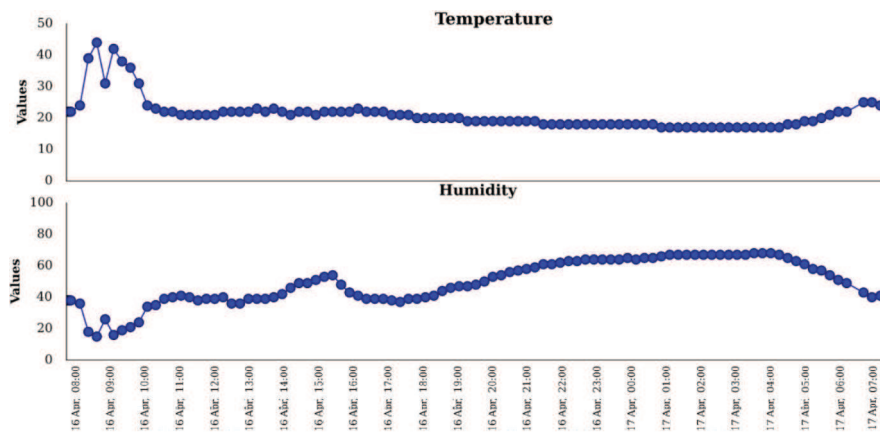


Figure 3.20: Temperature and relative humidity acquired in an outdoor environment during a single day.

3.5 EXCHANge: a solution to Key-Establishment for IoT leveraging Channel Anonymity

The previous chapters discussed KMP that leverage public key cryptography techniques, assuming that IoT devices are able to perform such operations, thorough hardware or software mechanisms.

However, there exist class of devices that cannot afford such computations, due to computational or storing constraints. Thus, other mechanisms are necessary to allow these devices to establish a shared key to be further used in symmetric cryptography operations (symmetric cryptography techniques are known to be more lightweight than asymmetric mechanisms).

In this context, physical layer security can provide a significant breakthrough, in particular to guarantee confidentiality among devices' message exchange. In fact, key-establishment can be performed by only exploiting the device's radio and the already existing communications among the devices, avoiding CPU expensive computations, battery drain and bandwidth overhead [117][118][119]. Physical layer security is recently gaining momentum, due to the resulting efficiency and effectiveness of the proposed solutions for wireless communications. In fact, this is the perfect solution for IoT scenarios, where the devices are always on and always connected, exchanging radio messages along their operational life-time.

Furthermore, the current state of the art and techniques previously discussed provides authentication (via a one-time process) while confidentiality is achieved by periodically refreshing the encryption key that has originally generated by the one-time process [120]. Such a solution is not secure: an adversary that is able to disclose all the device's secrets (e.g via cryptanalysis or just temporally-limited physical compromise) will have full access to all the subsequent communications.

A secure solution to key-establishment should not leverage any pre-existing secret stored in the device memory: although asymmetric encryption represents an effective solution to this problem, it is not the most efficient one because it introduces a significant CPU burden that cannot be tolerated in scenario where frequent re-keying operations are needed and there are tight CPU constraints. Moreover, there are even scenarios where the device architecture cannot simply implement the asymmetric

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

solution—for instance, in the low-end IoT devices [121]. An ideal solution should minimize the usage of the CPU, possibly exploiting the already existing communications, and avoiding any bandwidth overhead.

Authors in [118] introduced COKE, a key-establishment protocol leveraging radio channel anonymity. It enjoys the following features: (i) it needs only one secure hash computation per key; (ii) it does not resort to any pre-existing secret in the device memory; and, (iii) it allows two peers to converge to a secret key by only exploiting the underlying radio communications among these two devices. COKE has been subsequently extended to be deployed on a wide network [119]. Nevertheless, COKE has not yet been implemented on a real architecture, and its performance have not yet been measured in real scenarios.

In line with these premises, this section introduces the EXCHANge protocol. EXCHANge is inspired by COKE, albeit it presents several modifications in order to meet the constraints of technologies and real devices making up the IoT. To show its effectiveness, the EXCHANge protocol has been implemented in the OpenWSN protocol stack and tested on the OpenMote-CC2538 architecture. As can be confirmed by a deep review of the literature, EXCHANge is the first real-world implementation of a key-establishment protocol based on channel anonymity. The following subsections provide also the results of an extensive measurement campaign in several configuration scenarios considering both the security and the performance of the protocol.

The performance achieved by EXCHANge represents a milestone for the key-establishment protocols working at the physical layer: EXCHANge can set-up a new secret key of 80 bits in less than 10 seconds (on average) while requiring only a single hash computation. The security of the EXCHANge protocol against a global eavesdropper adversary strategically deployed in the playground have been also successfully tested. Finally, an experimental measurement of the energy consumption has been provided showing that the EXCHANge protocol is able to perform key-establishment consuming just the 0.14% of the battery capacity of a node, in the worst case scenario.

3.5.1 Scenario and adversary model

Figure 3.21 depicts the scenario considered in the following discussion: two nodes are assumed, namely $\mathcal{A}$ and $\mathcal{B}$, that want to establish a secret key K of length $2N$ by starting from two random secret seeds K_A and K_B (known only to A and B , respectively)

3.5 EXCHANge: a solution to Key-Establishment for IoT leveraging Channel Anonymity

of length N and without resorting to either pre-loaded secrets or Public Key Infrastructure (PKI) based solutions such as [49]. EXCHANge has minimal requirements to make two devices to establish a secret key: $\mathcal{A}$ and $\mathcal{B}$ have to feature a radio, being able to exchange each others radio messages, while devoting to the protocol minimal CPU power and memory space.

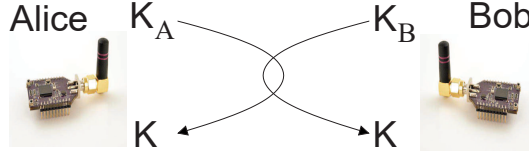


Figure 3.21: EXCHANge scenario.

The scenario envisages an adversary $\mathcal{E}$ provided with an omni-directional antenna able to eavesdrop all the communications from both the radio spectrum and the deployment scenario. The scenario also assumes that $\mathcal{E}$ is able to control the environment *before* the deployment of the devices and therefore being aware of all the transmitted messages by the involved devices.

3.5.2 Key sharing based on channel anonymity

It is assumed that both $\mathcal{A}$ and $\mathcal{B}$ have anonymized their radio messages by removing all the information that could lead to the identification of the transmitting source, e.g., by sanitizing their MAC addresses. Moreover, for each round of the protocol, the transmitter is randomly selected between $\mathcal{A}$ and $\mathcal{B}$, for instance, via a channel contention algorithm. Let's consider c as a random bit-vector of size $2N$ such that when $c[i] = 1$ the transmitting source is $\mathcal{A}$; otherwise, when $c[i] = 0$, the transmitting source is $\mathcal{B}$ — $i \in [0, 2N - 1]$. Therefore, the vector c controls the interleaving process of $\mathcal{A}$ and $\mathcal{B}$ on accessing the radio channel.

Let's now consider the toy example of Figure 3.22: $\mathcal{A}$ and $\mathcal{B}$ initiate the protocol by randomly choosing two secret keys $K_A = [0, 0]$ and $K_B = [1, 1]$, respectively. Also, let's assume $c = [1, 0, 1, 0]$, such that the peers interleave as $[\mathcal{A}, \mathcal{B}, \mathcal{A}, \mathcal{B}]$. Now, at each round $i \in [0, 3]$ of the protocol, the peers apply the following rule: if the elected node for the transmission is $\mathcal{A}$, then the extracted bit b from K_A is transmitted as complemented, otherwise b is transmitted as it is—note that this rule, being agreed in clear-text at run time, is known by $\mathcal{E}$ as well. In the example of Figure 3.22, $\mathcal{A}$ and $\mathcal{B}$

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

transmit the sequences $[1, 1]$ and $[1, 1]$ respectively, and converge to the shared secret key $K = [0, 1, 0, 1]$. The adversary $\mathcal{E}$, in the middle of the scene, cannot distinguish the transmitting source, and therefore it cannot make a decision if complementing or not the bit it has just eavesdropped. It is worth noticing that the security of the protocol does not rely on the value of the transmitted bit; it relies only on the *hidden identity of the source*: as soon as the anonymity of the source is protected, $\mathcal{E}$ cannot deterministically decide if the current bit has to be complemented or not, and therefore the secrecy of the transmitted bit is guaranteed—ideally, the best possible educated guess would be to randomly choosing the transmitter. Figure 3.23 wraps up on the four pos-

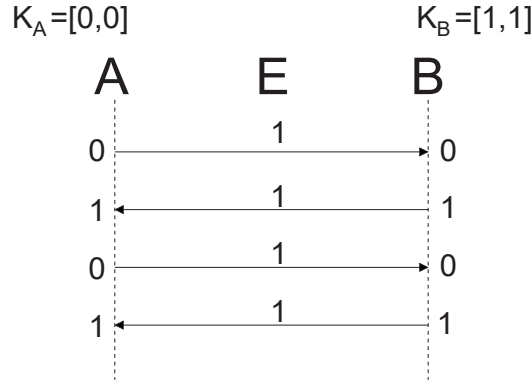


Figure 3.22: Toy example of the EXCHANGE protocol: $\mathcal{A}$ and $\mathcal{B}$, randomly choose a secret key, e.g., $K_A = [0, 0]$ and $K_B = [1, 1]$, and then they randomly interleave their transmissions: $[\mathcal{A}, \mathcal{B}, \mathcal{A}, \mathcal{B}]$. The transmitted bit is complemented when the transmitter is $\mathcal{A}$, while it is transmitted as it is when the transmitting source is $\mathcal{B}$.

sible scenarios characterizing the channel anonymity: each round is characterized by a transmission of one bit by either $\mathcal{A}$ or $\mathcal{B}$. If the transmitter is $\mathcal{A}$ than the transmitted bit is complemented, otherwise the bit is transmitted as it is.

Conversely $\mathcal{E}$, who wants to disclose the secret bit, implements the so called *RSS proximity based attack*: $\mathcal{E}$ can move closer to one of the peer to leverage the received signal strength (RSS) and infer on the actual identity of the source. As it will be clear in the following, EXCHANGE is robust to the above attack. Accordingly, the results of a test-bed has been provided, that clearly show the protocol’s efficiency and effectiveness.

3.5 EXCHANge: a solution to Key-Establishment for IoT leveraging Channel Anonymity

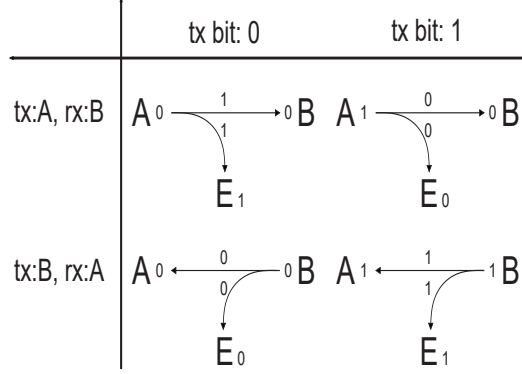


Figure 3.23: Channel anonymity: as soon as the anonymity of the transmitter is protected, $\mathcal{E}$ cannot deterministically decide whether to complement the received bit and therefore the secrecy of the transmitted bit is protected.

3.5.3 The EXCHANge key-establishment protocol

3.5.3.1 Definitions and assumptions

Let c be a vector of $2N$ random bits such that $c \xleftarrow[\$]{2N} [0, 1]$. The notation $c \xleftarrow[\$]{2N} [0, 1]$ is used to define the generation of $2N$ pseudo-random distinct elements from $[0, 1]$ and assign them to c . Therefore, $\mathcal{A}$ and $\mathcal{B}$ interleave their transmission in a mutually exclusive fashion, i.e., if $c[i] = 1$ then $\mathcal{A}$ is the transmitting source, otherwise ($c[i] = 0$) $\mathcal{B}$ will transmit the message. Nevertheless, due to the distributed nature of the transmitting source selection, there might be a few cases for which $\mathcal{A}$ and $\mathcal{B}$ transmissions collide: Section 3.5.3.5 specifically considers this case.

Let $K_A \xleftarrow[\$]{N} [0, 1]$ and $K_B \xleftarrow[\$]{N} [0, 1]$ be the secret seeds generated by $\mathcal{A}$ and $\mathcal{B}$, respectively, and being constituted by a sequence N of pseudo-random bits.

Definition 3.5.1 Let $K_A[a(i)]$ ($K_B[b(i)]$) be the bit to be transmitted by $\mathcal{A}$ ($\mathcal{B}$) at time slot i , with:

$$a(i) = \sum_{j=0}^i c[j] - 1, \quad b(i) = \sum_{j=0}^i \bar{c}[j] - 1, \quad \text{with } i \in [0, 2N - 1], \quad (3.15)$$

where the notation $\bar{x}$ is used to denote the binary complement of x .

Definition 3.5.2 Let $e[i]$ be the encrypted bit transmitted during the time slot i , yielding:

$$e[i] = \begin{cases} \bar{K}_A[a(i)] & \text{if } \mathcal{A} \text{ is transmitting} \\ K_B[b(i)] & \text{if } \mathcal{B} \text{ is transmitting.} \end{cases} \quad (3.16)$$

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

Definition 3.5.3 Let $K \in [0, 1]^{2^N}$ be the shared secret key generated by the EXCHANge protocol, and $K[i]$ be the shared bit-string at time slot i , yielding:

$$K[i] = c[i] \oplus e[i]. \quad (3.17)$$

Therefore, each transmitted bit is encrypted with the identity of the source. As an example, recalling Figure 3.22, $c = [1, 0, 1, 0]$, $K_A = [0, 0]$, $K_B = [1, 1]$, $e = [1, 1, 1, 1]$, and $K = [0, 1, 0, 1]$. From Eq. (3.17), it is possible to observe that the value of the current transmitted bit $e[i]$ does not leak any useful information if the anonymity of the transmitting source is secret, i.e., if $c[i]$ is unknown to $\mathcal{E}$. It is worth noticing the difference among the following terms: $K_A[a(i)]$ ($K_B[b(i)]$) is the bit pseudo-randomly generated by $\mathcal{A}$ ($\mathcal{B}$), $e[i]$ is the encrypted bit transmitted at time slot i , and finally $K[i]$ is the shared bit generated at time slot i .

In order to deploy the EXCHANge protocol in a real IoT scenario, it is necessary to modify three fields of the IEEE 802.15.4 radio message [22]. Firstly, as previously introduced, both the source and the destination addresses should be sanitized: it is possible to set them to $00:00:00:00:00:00$. Lastly, the encrypted bit $e[i]$ is stored in the least significant bit (LSB) of the MAC payload of the message. Figure 3.24 shows the details about the information injected by the EXCHANge protocol in a IEEE 802.15.4 radio message.

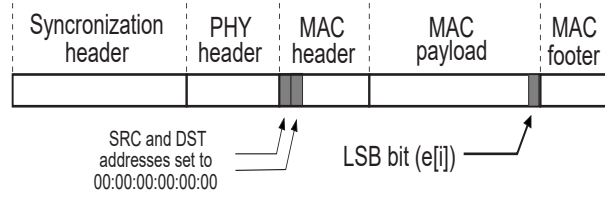


Figure 3.24: EXCHANge protocol: packet format for the IEEE 802.15.4e standard. The source and destination addresses have been sanitized, i.e., set all to zeros, and the least significant bit (LSB) of the MAC body is set accordingly to Eq. (3.16).

3.5.3.2 Finite state machine model

The following discussion provides the details of a finite state machine (FSM) model of the EXCHANge protocol. The FSM model has been designed with the aim of integrating the EXCHANge protocol within the TSCH mode of the IEEE 802.15.4 standard.

3.5 EXCHANGE: a solution to Key-Establishment for IoT leveraging Channel Anonymity

The protocol is characterized by three radio operative modes: *TX*, *RX*, and *SLEEP*. During the *SLEEP* mode the radio is inactive, while during the *TX* and the *RX* modes the radio is on, transmitting and receiving, respectively. Each of the status identified within the FSM performs a specific task. In detail:

- **PREP_TX.** A new message m is set up according to the IEEE 802.15.4e standard. Assuming the current slot as i , the LSB value of the MAC payload is set to the value of the encrypted bit $e[i]$, the message m is loaded into the radio buffer, the frequency is set and a random transmission power is chosen.
- **RAND_GEN.** A random number $\delta \xleftarrow{\$} [0, \Delta]$ is generated, Δ being a system parameter.
- **RAND_WAIT.** The node sets up a timer and waits a portion of time related with δ before initiating the transmission. While waiting, the node sets up its radio in the *RX* status. If no message is received before the expiration of the timer, then the node proceeds with the next status.
- **TX_GO.** The radio is switched between the *RX* and the *TX* status for initiating a new transmission.
- **TX_DATA.** The message m is transmitted to the peer.
- **RX_DATA.** The message m is received from the peer.
- **POST_TX.** The message m is further elaborated after the transmission (i.e., the LSB bit of the message payload is recorded and complemented (if needed), and local registers are cleared).
- **PARSE_RX.** The receiving node has now completed the reception of the message m . Therefore, the LSB of the message m , i.e., the encrypted bit $e[i]$, is extracted from the transmitted message m , decrypted and added to the shared secret key K .
- **PREP_ACK.** A new *ack* message is set up with a sequence number consistent with the last received message, the frequency is set and a random transmission power is chosen.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

- **WAIT_ACK**. The node that sent the message waits for the peer to setting up the *ack* message.
- **TX_ACK (RX_ACK)** . The transmitting (receiving) node transmits (receives) the *ack* related to the last received message.
- **PARSE_ACK**. Processing of the received *ack* message.
- **RX_PROC**. Post-processing of the *ack* message just delivered.

3.5.3.3 The EXCHANge algorithm

The details of the EXCHANge protocol are depicted in Algorithm 1, with reference to the statuses previously introduced. The *send* calls are non-blocking, while the *receive* calls are blocking and can be resumed by an external event when the associated boolean variable becomes true, i.e., depicted as $\downarrow$ *variable* in Algorithm 1. The end of the slot clears and resets the protocol parameters and variables, independently of its current status (line 24). Each of the node starts the protocol by choosing a random bit string, i.e., K_A for $\mathcal{A}$ and K_B for $\mathcal{B}$. At each instance i of the protocol, a random transmission power is selected (line 5) and a random waiting time—for a new message reception—is scheduled (lines 6–7). When the waiting time expires, the node initiates a new transmission of a message carrying a new secret bit: the bit is transmitted as complemented if $\mathcal{A}$ is the source, while it is transmitted as it is if the source is $\mathcal{B}$ (lines 8–12). Conversely, the peer receives, complements and stores the bit accordingly (lines 7-14). Finally, the receiving node acknowledges the last received bit by sending back to the transmitter the *ack* message with the same sequence number $sn[i]$ of the transmitted message (line 15). The peer receives the *ack* message and adds the secret bit to its own shared secret key (line 19). The sequence number is verified and if consistent a new bit is considered for transmission (line 22), otherwise the current bit is re-transmitted. Finally, the bit-string K is hashed by cryptographically secure hash function $H(\cdot)$ (lines 26-28).

3.5.3.4 Implementation of the FSM

The FSM illustrated in Section 3.5.3.2 has been implemented by using the OpenMote-CC2538 described in Section 1.4.1.3.

3.5 EXCHANge: a solution to Key-Establishment for IoT leveraging Channel Anonymity

Algorithm 1: EXCHANge protocol

```

1  let  $K_A \xleftarrow{\$} [0, 1]$  and  $K_B \xleftarrow{\$} [0, 1]$ .
2  let  $p_m$  and  $p_M$  be the minimum and maximum transmission powers, respectively.
3  let  $sn[i]$  be the sequence number of the current message;
4  let  $2N$  be the size of the secret key;
5  let  $H(\cdot)$  be a cryptographic secure hash function;
6
   /* PREP_TX */
   /* The peer randomly chooses the transmission power */
7   $T \xleftarrow{\$} [p_m, p_M]$ ;
   /* Select a random waiting time: ELAPSED is True when such time is elapsed,
      False otherwise. */
   /* RAND_GEN - RAND_WAIT - RX_DATA */
8  ELAPSED = FALSE;
9  receive( $e[i], sn[i]$ ); // ↓ ELAPSED
   /* TX_DATA - POST_TX */
10 if ELAPSED then
    /* Send the  $i^{th}$  bit with seq. number  $sn[i]$  */
11     $e[i] = \overline{K}_A[a(i)]$ ; /* if  $\mathcal{B}$  :  $e[i] = K_B[b(i)]$ ; */
12     $sn = sn[i]$ ;
13    send( $e[i], sn[i]$ );
14 end
15 else
    /* PARSE_RX */
    /* Extract the secret bit from the received message */
16     $K[i] = \overline{e}[i]$ ; /* if  $\mathcal{B}$  :  $K[i] = e[i]$ ; */
    /* PREP_ACK - TX_ACK */
    /* Transmit the ack */
17    send( $sn[i]$ );
18 end
19 if ELAPSED then
    /* WAIT_ACK - RX_ACK */
    /* Receive the ack */
20    receive( $sn[i]$ );
    /* Add a new bit to the key */
21     $K[i] = \overline{e}[i]$ ; /* if  $\mathcal{B}$  :  $K[i] = e[i]$ ; */
22 end
    /* PARSE_ACK - RX_PROC */
23 if  $sn[i] == sn$  then
24     $i = i + 1$ ;
25 end
26 if  $i \geq 2N$  then
    /* Generate a secret key. */
27     $K = H(K)$ ;
28 end
    /* Wait for the current slot to expire. */
29 RESET;
```

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

As previously explained, each slot of the IEEE 802.15.4 communication protocol lasts for 10ms, equivalent to 329 32-KHz radio tics. Therefore, the duration of each status in the FSM has been tuned in order both to meet this high-level requirements about the duration of the whole slot, and carefully considering the speed of execution of the CC2538 SoC. The result of this tuning operation is depicted in Figure 3.25. Moreover,

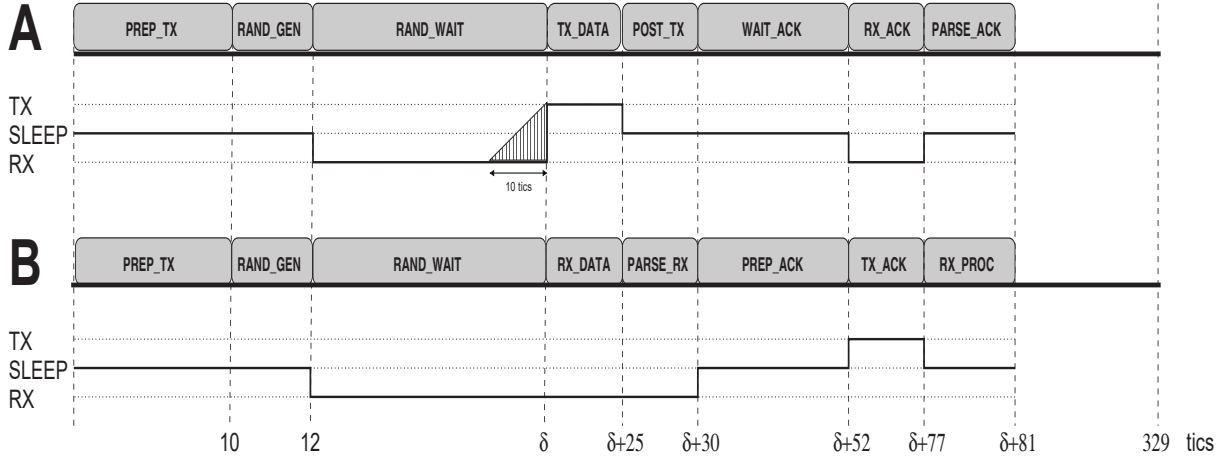


Figure 3.25: EXCHANGE timeline: both $\mathcal{A}$ and $\mathcal{B}$ start in *sleep* mode, and switch to *RX* mode after 12 tics. Subsequently, both $\mathcal{A}$ and $\mathcal{B}$ schedule a random wait period: the first one ($\mathcal{A}$) reaching the period's end, it switches to the *TX* status initiating the transmission. Conversely, the second one ($\mathcal{B}$) remaining in the *RX* status, it receives the message. Eventually, the receiver ($\mathcal{B}$) acknowledges the packet reception with an *ack* message.

the duration of each status of the FSM is included in Table 3.7. As for the random number extraction process and the *RAND_WAIT* status, it is worth noting that only natural integers numbers can be extracted, and that the upper bound of the random wait needs to be carefully set in order not to exceed the time-slot duration of 10ms. Moreover, a further restriction is imposed by the time necessary to switch the radio between the *RX* and the *TX* mode, that is 10 tics. Given all these considerations, 23 possible choices are allowed.

3.5.3.5 Collisions and Lost packets

This section details the features that make the EXCHANGE protocol robust to collisions and packet loss.

Collisions. A collision event might happen when both $\mathcal{A}$ and $\mathcal{B}$ choose the same

3.5 EXCHANGE: a solution to Key-Establishment for IoT leveraging Channel Anonymity

Table 3.7: EXCHANGE: Statuses of the finite state machine.

Status name	Duration (tics)
PREP_TX	10
RAND_GEN	12
RAND_WAIT	$[0, \Delta]$
TX_DATA	25
RX_DATA	25
POST_TX	5
PARSE_RX	5
WAIT_ACK	22
PREP_ACK	22
RX_ACK	25
TX_ACK	25
PARSE_ACK	4
RX_PROC	4

random wait time. Moreover, recalling Figure 3.25, it is possible to observe that during the random wait, a node switching from the *RX* to the *TX* status needs $\epsilon_s = 10$ tics: this period of time is also a source for potential collisions, e.g., the nodes might switch concurrently and eventually collide. The probability of collision can be computed by observing that the **RAND_WAIT** period is a uniformly distributed random variable in the interval $[0, \Delta]$, with $\Delta = 236$, yielding:

$$P_C = \frac{1}{\lfloor \Delta / \epsilon_s \rfloor} = \frac{1}{23} \approx 0.04. \quad (3.18)$$

When $\mathcal{A}$ and $\mathcal{B}$ collide for the transmission at time slot i^{th} , the *ack* message is not generated and eventually the protocol is instantiated again for the establishment of the same bit.

Packet loss. Wireless communications are prone to packet losses due to multipath fading and low signal-to-noise ratio. The protocol is robust to packet loss: the *ack* transmitted for each of the received bit guaranteeing that a new bit is added to the shared secret key only when both the peers have agreed on it. If either the message or the *ack* are lost during the i^{th} run of the protocol, no bit is added to the shared secret key and the protocol is instantiated again for the establishment of the i^{th} bit.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

3.5.4 Performance

This section provides the results of a measurement campaign characterized by a benign scenario, i.e., without the presence of any adversary. Indeed, this section provides details about the performance of the EXCHANge protocol in terms of robustness to packet loss and required time to exchange a bit-string of a given length. At the same time several concepts that will be used when testing the security of EXCHANge against a global eavesdropper adversary are introduced.

Firstly, from subsection 3.5.3.2 it is possible to recall that each instance of the EXCHANge protocol allows to generate (at most) one shared bit and that it has to commit inside a single slot of the IEEE 802.15.4 protocol, lasting for 10ms. This means that, in order to generate 128 secret bits, the protocol needs at least 128×10 ms equal to 1.28s. Moreover, it must be also taken into account that possible collision events introduce (on average) a 4% overhead, thus yielding to 1.33s (see Section 3.5.3.5). Therefore, an optimistic estimation for the EXCHANge protocol shows that it needs 1.33s to generate a shared bit-string of 128 bits.

Figure 3.26 shows the time to generate a secret key of 128 bits as a function of

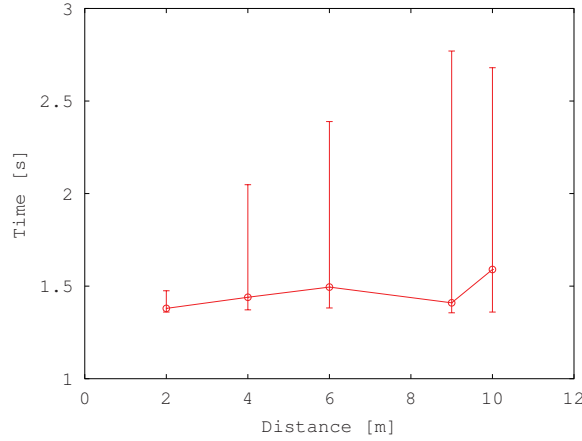


Figure 3.26: Time to run the EXCHANge protocol and generating a shared bit-string of 128 bits as a function of the distance between the devices. The errorbars show the quantile 5, 50, and 95 related to 20 runs of the protocol.

the distance between $\mathcal{A}$ and $\mathcal{B}$. Increasing the distance between $\mathcal{A}$ and $\mathcal{B}$ introduces a significant packet loss that in turn affects the time to establish the secret key. In particular, for long distances (≥ 8 m) it is possible to observe a high variance in the

3.5 EXCHANge: a solution to Key-Establishment for IoT leveraging Channel Anonymity

key-establishment process due to the path loss and moving objects in the peers' surroundings. Finally, on average, the protocol needs about 140 slots (about 1.4 seconds) to exchange 128 bits.

The slot-frame size. The performances of the EXCHANge protocol also depend on the configuration parameters of the IEEE 802.15.4 protocol. In fact, the slots of the IEEE 802.15.4 protocol (that can be actually used to run the EXCHANge protocol) are grouped into slot-frames. The number of slots belonging to a slot-frame can be varied freely, according to the duty-cycle required for a given application and throughput considerations. The tests assume a slot-frame size varying between 11 and 101, where 11 is the default value of the slot-frame size provided by the OpenWSN protocol stack that runs on the OpenMote-CC2538, while 101 is the slot-frame size recommended by the reference document [33] and widely used in literature and also in experiments described in the previous sections of this chapter [50, 90]. Therefore, recalling from the above section an average time $oft_p = 1.4$ seconds to generate a shared bit-string of 128 bits, and assuming the slot-frame as constituted by $N_{SF} \in [11, \dots, 101]$ slots, the time t_P needed by the EXCHANge protocol to generate a bit-string when N_P out of N_{SF} frames are dedicated to it, yields:

$$t_P = \left\lceil t_p \cdot \frac{N_{SF}}{N_P} \right\rceil$$

. The above equation is depicted by Figure 3.27, showing the time (in seconds) necessary to generate a bit-string of 128 bits, assuming a IEEE 802.15.4 slot-frame of a given size, while varying the number of slots dedicated to the key establishment protocol.

In general, the more the number of slots dedicated to the key agreement protocol, the faster the protocol will generate a key. Observe that, given a slot-frame of size N_{SF} , the higher number of slots that can be dedicated to the protocol is $N_{SF} - 1$, because at least one slot is needed for guaranteeing standardized communication and tight synchronization between neighboring nodes. As anticipated before, the network administrator can choose the slotframe size and the number of active slots starting from considerations on the required duty-cycle and network throughput. Finally, it is possible to observe from Figure 3.27 that choosing a slot-frame size of $N_{SF} = 101$ slots with $N_P = 100$ slots dedicated to the EXCHANge protocol allows to generate a bit-string in about 1.42 seconds—that is pretty close to the optimal time that can be reached when considering a bit-string of 128 bits.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

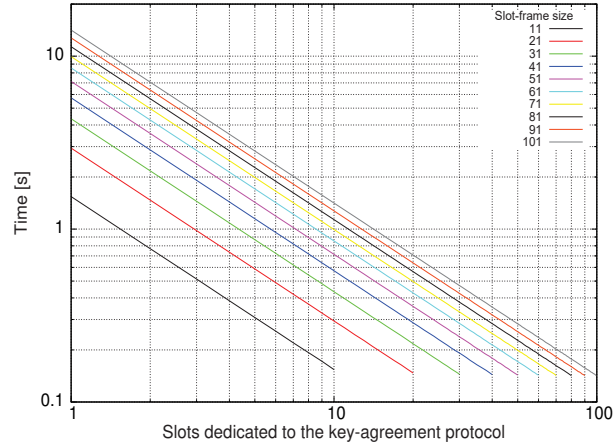


Figure 3.27: Time to generate a bit-string of 128 bits varying the number of slots dedicated to the EXCHANGE protocol in the slot-frame. A slot-frame size spanning in the interval $[11, \dots, 101]$ is assumed.

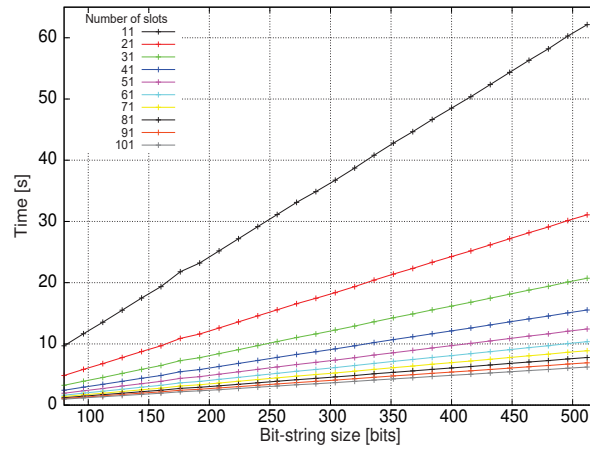


Figure 3.28: Time to transmit a bit-string of different sizes varying the number of dedicated slots (between 1 up to 10) in a slot-frame of 11 slots (default value for the OpenWSN protocol stack).

3.5 EXCHANge: a solution to Key-Establishment for IoT leveraging Channel Anonymity

Depending on the security requirements of the application, the EXCHANge protocol allows the two peers to generate a shared secret bit-string of arbitrary size. Figure 3.28 shows the average time needed to generate a bit-string of size spanning between 80 up to 1000 bits while varying the number of dedicated slots N_P of the EXCHANge protocol in a slot-frame of $N_{SF} = 11$ slots.

3.5.5 Security Analysis: guessing the secret key

The adversary considered in this section is fully passive: $\mathcal{E}$ can eavesdrop and log all the communications happening on all the frequencies of the radio spectrum and throughout the playground. Moreover, $\mathcal{E}$'s goal is to disclose the secret key that $\mathcal{A}$ and $\mathcal{B}$ generate during the consecutive runs of the EXCHANge protocol. To achieve her goal, the best strategy for $\mathcal{E}$ is to eavesdrop the transmitted messages and trying to infer on the transmitting source. In fact, recalling Eq. (3.17) the transmitted bit $e[i]$ is encrypted with the identity of the source $c[i]$, and therefore the value of the encrypted bit $e[i]$ does not leak any information about the current value of the transmitted bit $K[i]$.

Assuming $2N$ successful and secret transmissions yielding to a key length of $2N$ bits, the key space is given by:

$$P_D = 2^{-2N}. \quad (3.19)$$

In order to successfully disclose the secret key with 50% probability by random guessing, $\mathcal{E}$ has to perform on average 2^{2N-1} trials, that is assumed to be computationally unfeasible when $2N \geq 80$. However, there can be cleverer attacks that can be played by $\mathcal{E}$ with the most dreadful one described in the following.

RSSI-based source identification. $\mathcal{E}$ can infer on the current source identity by measuring the received signal strength (RSSI), and performing a proximity estimation, knowing the devices' deployment. Figure 3.29 shows the RSSI-based attack implemented by the adversary. Assuming $\mathcal{E}$ is aware of the devices' layout, it can move closer to one of the two device (Figure 3.29(a) and (c)), hence being able to identify the transmitting source by performing an educated guess based on the received signal strength. Figure 3.30 shows the probability mass function associated to the received signal strength measured by $\mathcal{E}$ at the three different positions of Figure 3.29. $\mathcal{E}$ experiences higher RSSI values from the closer device while she measures lower signal

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

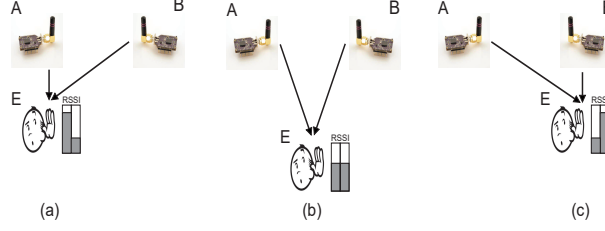


Figure 3.29: RSSI-based source identification: $\mathcal{E}$ measures the received signal strengths R_A and R_B from $\mathcal{A}$ and $\mathcal{B}$, respectively, and infers on the current transmitting source by looking at the different levels of the received power: (a) $R_A \geq R_B$, $R_A \approx R_B$, and $R_A \leq R_B$.

strengths from the farther device: it is also worth noticing that RSSI values in Figure 3.30(a) and Figure 3.30(c) can be parted into two distinct sets corresponding to the transmitted messages by $\mathcal{A}$ and $\mathcal{B}$. Conversely, $\mathcal{E}$'s guessing capabilities are significantly reduced when the received signal levels from $\mathcal{A}$ and $\mathcal{B}$ have similar values (Figure 3.29(b) and Figure 3.30(b)). The toy example of Figure 3.29 assumes both $\mathcal{A}$ and $\mathcal{B}$ are transmitting with the same power. That is why EXCHANge uses random transmission power. In particular, recalling Algorithm 1 (line 7), the EXCHANge protocol mitigates the RSSI-based source identification by *randomizing the transmitting power*. Under such an assumption, i.e., random modulation of the transmission power, $\mathcal{A}$ and $\mathcal{B}$ can mimic different distances to $\mathcal{E}$, and therefore, by shuffling their respective “fake” positions, they regain the source anonymity.

3.5.5.1 Mitigating the RSSI-based source identification

Source identification can be mitigated by exploiting random transmission power, as in the EXCHANge protocol: since $\mathcal{E}$'s guessing is only based on the received signal strength, $\mathcal{A}$ and $\mathcal{B}$ might pretend to switch their positions at each round of the EXCHANge protocol by enforcing random transmission powers. Moreover, moving objects in the line-of-sight between $\mathcal{E}$ and the peers can drastically change the received signal strength experienced by $\mathcal{E}$ and therefore making her guessing even more challenging. This section evaluates the performance of the EXCHANge protocol in a real world scenario where $\mathcal{E}$ can get closer to a target node to increase her chances to discriminate the transmitting source by looking at the received signal strength and therefore guessing the values of the generated bits. Assume that $\mathcal{A}$ and $\mathcal{B}$ can randomly change their

3.5 EXCHANGE: a solution to Key-Establishment for IoT leveraging Channel Anonymity

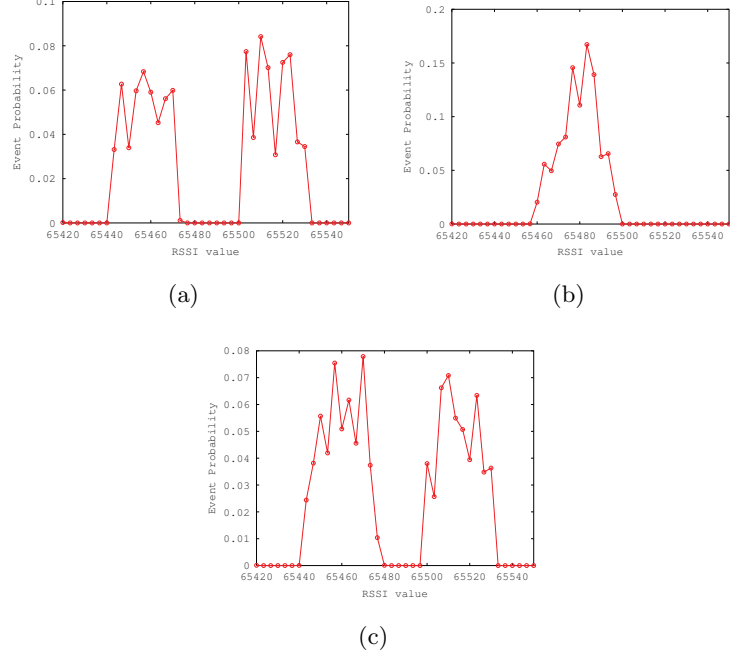


Figure 3.30: Probability mass function associated to the received signal strengths experienced by $\mathcal{E}$ during real measurements featuring the scenarios of Figure 3.29.

transmission power and the playground is affected by moving objects such as people moving around and cutting the line-of-sight between $\mathcal{A}$ and $\mathcal{B}$.

Figure 3.31 shows $\mathcal{A}$ and $\mathcal{B}$ deployed at 6 m of distance from each other and 0.8 m of height from the ground, in an open area clear from obstructions and radio interferences. Then, 4 different positions are considered for $\mathcal{E}$ sitting at 3, 2, 1, and 0.5 m to $\mathcal{A}$, respectively. For each deployment, 20 bit-string of 128 bits have been transmitted, for a total of about 2560 exchanged bits.

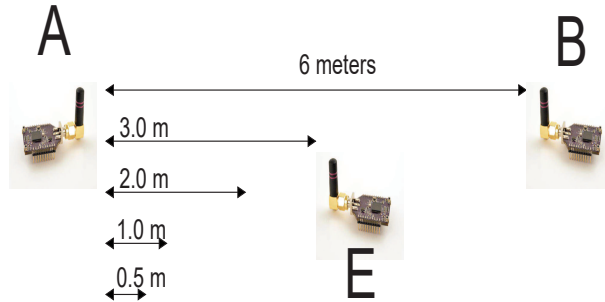


Figure 3.31: Experimental set-up: $\mathcal{A}$ and $\mathcal{B}$ are placed at 6 meters far away while $\mathcal{E}$ is moved from 3m up to 0.5m close to $\mathcal{A}$.

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

The best strategy to identify the actual transmitting source S relies on an educated guess that maps the current received signal strength R to the identity of the transmitter (either $\mathcal{A}$ or $\mathcal{B}$). In particular, it is well known that the received signal strength becomes weaker when the distance between the transmitter and the receiver becomes larger [122]. Therefore, $\mathcal{E}$'s best strategy is to define a received signal strength threshold τ and assuming that the signals received with a strength exceeding the threshold τ have been transmitted from the closer device ($\mathcal{A}$), while those ones experiencing a received signal strength below the threshold τ have been transmitted by the farther device, yielding:

$$S = \begin{cases} \mathcal{A} & \text{if } R \geq \tau, \\ \mathcal{B} & \text{if } R < \tau. \end{cases}$$

The value of the threshold τ can be empirically evaluated by considering the average value of all the possible received signal strengths for a specific scenario [118], yielding:

$$\tau = R_{min} + \frac{R_{max} - R_{min}}{2},$$

where R_{max} and R_{min} are the maximum and minimum RSS values experienced by $\mathcal{E}$ for a specific deployment scenario.

As an example of the above procedure, Figure 3.32 refers to the scenario of Figure 3.31 where $\mathcal{E}$ is at a distance of 1.5m from $\mathcal{A}$. Black lines shows the probability mass function (p.m.f.) associated to the received signal strengths experienced by $\mathcal{E}$ while the red line shows the threshold ($\tau = 65459$) computed by the adversary to discriminate between the sources. $\mathcal{E}$ will decide that the signals with received signal strength over the threshold have been transmitted by $\mathcal{A}$ while the others have been transmitted by $\mathcal{B}$; she will change the values of the previously eavesdropped bit accordingly—we recall that the bits transmitted by $\mathcal{A}$ have to be complemented.

Now, for each of the $\mathcal{E}$'s positions depicted in Figure 3.31, it is possible to count the number of secret bits, i.e., not-guessed by $\mathcal{E}$ applying the above procedure. Figure 3.33 shows the quantile 5, 50, and 95 computed over the number of bits not-guessed by the adversary in a sequence of 20 bit-string of 128 bits length each, as function of the position of $\mathcal{E}$: 0.5, 1, 2, and 3m to $\mathcal{A}$. The number of bits that continues being kept secrets increases when $\mathcal{E}$ moves farther away from $\mathcal{A}$: indeed, $\mathcal{E}$ decreases her chances to guess the correct values of the bits when she moves away from the target node ($\mathcal{A}$) up to her worst case positioning (in the middle of the scene), 3m away from each device; in

3.5 EXCHANGE: a solution to Key-Establishment for IoT leveraging Channel Anonymity

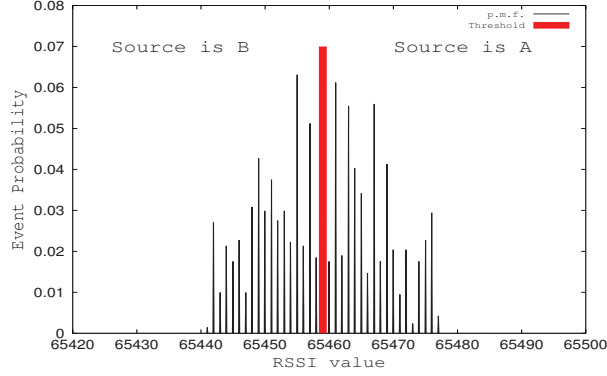


Figure 3.32: RSSI-based source identification: $\mathcal{E}$ splits the set of experienced RSS values into two subsets. Then, she decides that the smaller values (on the left of the red bar) are from the farther device ($\mathcal{B}$) while the bigger ones (on the right of the red bar) are from the closer node ($\mathcal{A}$).

this location her guessing capabilities are just slightly better than random guess, i.e., about 52 bits on average (the rightmost point of the curve in Figure 13) out of 128 bits. Moreover, we observe that the variance of the secret bits tends to reduce for greater distances: these phenomena can be explained by resorting to environmental factors. Indeed, when $\mathcal{E}$ is very close to $\mathcal{A}$ (0.5 meters), people moving around contribute either constructively or destructively to the guessing process introducing a significant variance in the guessing process. Conversely, when $\mathcal{E}$ moves farther away from $\mathcal{A}$, the received signal strength is more stable due the longer distances between the nodes and the moving objects and therefore the variance is significantly tighter.

It is worth noting that, for $\mathcal{E}$, the information implied from Figure 3.33 can have just a statistical significance—however, their importance is not negligible, as reported in the following. Indeed, assuming a conservative stance, such as $\mathcal{E}$ is able to derive the same data reported in Figure 3.33 and that she is at 1m from $\mathcal{A}$. From Figure 3.33, she knows that (on the average) 48 bits of the keys are kept secret, while 80 have been correctly guessed. However, she is not aware of which bits are the correct ones and which ones have to be flipped to derive the correct key—the information derived from Figure 3.33 is purely a statistical one. All she knows is that, on average, she has to make 2^{47} decryption attempts over a single eavesdropped encrypted bit-string to recover the secret key with 50% probability.

Starting from the above results, the number of transmissions has been computed in

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

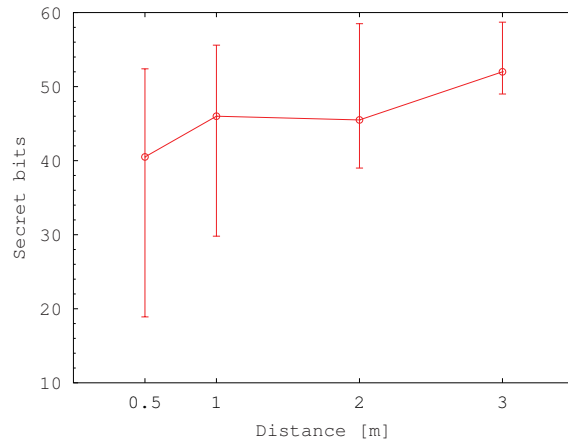


Figure 3.33: $\mathcal{E}$ guessing capabilities: quantile 5, 50, and 95 associated to the number of bits not-guessed by $\mathcal{E}$ (secret bits) during the RSSI-based source identification attack on a bit-string of 128 bit varying the distance between $\mathcal{E}$ and $\mathcal{A}$ as depicted in Figure 3.31.

order to guarantee 80 secrets bits, i.e., the number of secret bits needed to meet the security level in today's wireless communications [137]. The lower bounds (best case for $\mathcal{E}$) have been considered from Figure 3.33 and the number of requested transmissions in order to obtain 128 secret bits have been evaluated. Figure 3.34 shows the number

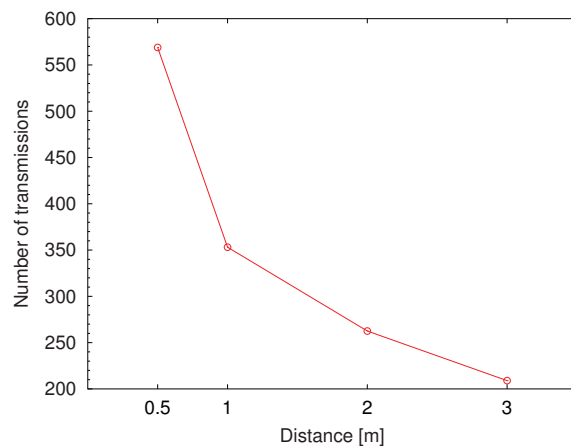


Figure 3.34: Number of maximum transmissions requested to establish at least 80 secret bits as a function of the distance between $\mathcal{A}$ and $\mathcal{E}$.

of transmissions requested to generate at least 80 secret bits as function of the distance between $\mathcal{A}$ and $\mathcal{E}$. The EXCHANGE protocol requests about 209 transmissions when $\mathcal{E}$ is far away from the devices, i.e., when the distance is greater than 3 meters, while it

3.5 EXCHANge: a solution to Key-Establishment for IoT leveraging Channel Anonymity

needs an increasing amount of transmissions when $\mathcal{E}$ moves closer to the target node: we measured about 569 transmissions (on average) to establish a secret key of (at least) 80 bits when $\mathcal{E}$ is only 50 cm far away from the target node ($\mathcal{A}$).

3.5.6 Energy analysis

This section introduces the methodology and the results of our analysis to measure the energy consumption of the EXCHANge protocol.

To measure the current drawn by the EXCHANge protocol a RIGOL DS1052E digital oscilloscope has been used, by sampling the voltage to the terminals of a 1Ω probe resistor connected to the current sense port of the OpenMote OpenBase. The RIGOL DS1052E has a vertical resolution of 8 bits, the vertical range has been set to 10mV/div while the horizontal range has been set to 2ms/div. Several runs of the protocol have been acquired and the data have been subsequently exported to GNU Octave for the analysis. Figure 3.35 shows the set-up to measure the energy consumption of the EXCHANge protocol.

Firstly, one run of the EXCHANge protocol has been identified and isolated as depicted in Figure 3.36. By comparing Figure 3.36 with Figure 3.25, it can be observed as the

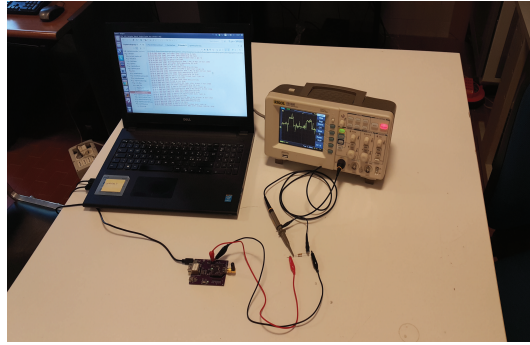


Figure 3.35: Set-up to measure the energy consumption.

device initially behaves as $\mathcal{A}$, transmitting one bit of the bit-string, and subsequently it behaves as $\mathcal{B}$, receiving another bit. The final transmission represents the *ack* message transmitted to acknowledge the last received bit. Figure 3.36 also depicts the current drain of the protocol during the three different statuses. It is possible to identify three different current drain levels, each one corresponding to a different device status, i.e.,

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

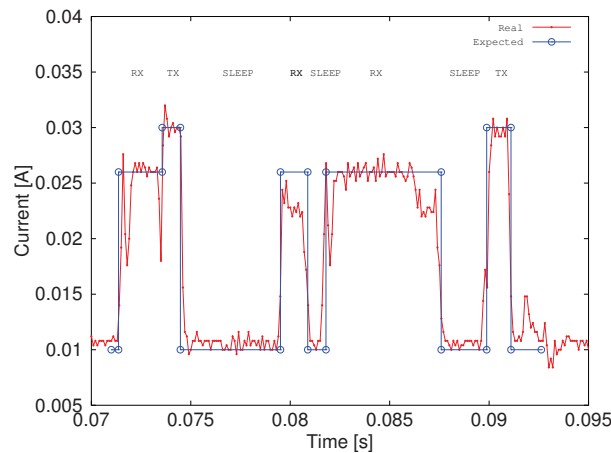


Figure 3.36: Current drain of one run of the EXCHANGE protocol: the device initially behaves as a transmitter and subsequently as a receiver.

SLEEP, *RX*, and finally *TX*. In order to precisely estimate the current drains, a statistical analysis have been performed about the current drain over all the collected measurements. Figure 3.37 shows the probability mass function associated to the current drain values sampled during extensive runs of the EXCHANGE protocol. Three major peaks can be observed: about 10mA when the device is in the *SLEEP* status, about 26mA when the device is in the *RX* status, and finally, about 30mA when the device is in the *TX* status. The above values are also confirmed by the OpenMote-CC2538 data-sheet [123]. Finally, it is possible to estimate the energy consumption of

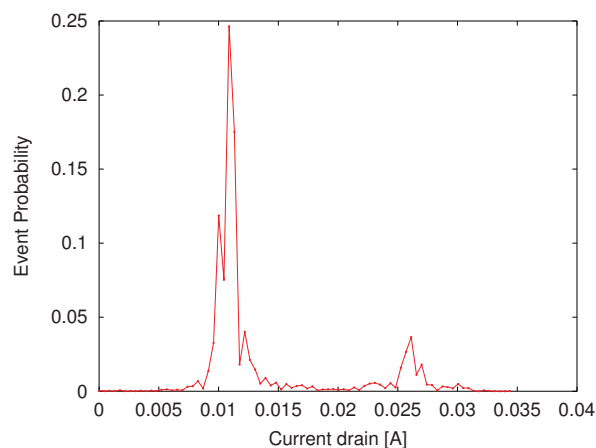


Figure 3.37: Probability mass function associated to the current drain sampled during multiple runs of the EXCHANGE protocol.

3.5 EXCHANge: a solution to Key-Establishment for IoT leveraging Channel Anonymity

% of battery lifetime	0.14	0.09	0.07	0.06
$\mathcal{A}$ - $\mathcal{E}$ distance [m]	0.5	1	2	3

Table 3.8: Protocol consumption assuming a secret key of 80 bits and estimation of the battery life percentage consumed by one run of the protocol assuming the mote is powered by two AA batteries.

the EXCHANge protocol by considering the time spent by the device in each of the status. During the measurements, any significant differences in the current drain was noticed when changing the transmission power, and therefore it has been assumed as constant and equal to 30mA the current consumption when the device is in the *TX* status.

The energy consumption can be computed by integrating the instantaneous current drain $i(t)$ over the time and multiplying it by 3.3V, i.e., the voltage of the OpenMote OpenBase board [123], yielding:

$$E[\text{mJ}] = 3.3\text{V} \int_{-\infty}^{\infty} i(t)dt. \quad (3.20)$$

Evaluating Eq. (3.20) over the values extracted from Figure 3.37 yields to an average power consumption of about 45 mJ. This value is particularly significant if compared with the consumption of the radio during the *SLEEP*, *TX* and *RX* modes, i.e., about 33, 99 and 86mJ by recalling Figure 3.36. Therefore, recalling the results from Figure 3.34, and assuming a consumption of 45mJ per exchanged bit, Table 3.8 resumes the major trade-offs of the protocol EXCHANge assuming at least 80 secret bits and a typical battery storage capacity of 9000J (AA-type)[124].

3.5.6.1 Discussion and Remarks

Key entropy. Contrary to other approaches, such as either RSSI or CSI-based key-establishments, the entropy of the key generated by the EXCHANge protocol does not strictly depend on multipath fading conditions. Other solutions such as [125][126][127] exploit temporal fluctuations of either the received signal strength or the channel state to generate secret bits: unfortunately, the wireless channel can be static for long periods, hence significantly delaying the key-establishment process—it has to wait for channel conditions with sufficient entropy. EXCHANge exploits the randomness of the internal pseudo-random number generator. Hence, it does not rely on any external

3. ADVANCED MAC-LAYER SECURITY FUNCTIONALITIES FOR LARGE-SCALE IOT NETWORKS

source of entropy, achieving performance that are independent of the channel conditions.

Resource constraints. It has been shown as the EXCHANge protocol is particularly lightweight in terms of both memory occupancy, CPU burden, and bandwidth. In particular, considering the CC2538 architecture, it requires just 4356 Bytes and 44 Bytes of ROM and RAM, respectively, and only one hash computation. As for the bandwidth, we showed that EXCHANge can be run by using only one out of the 11 slots in the IEEE 802.15.4 protocol, while the required energy to establish a secret key of at least 80 bits requires the 0.14% of the battery capacity in the worst case scenario, i.e., when $\mathcal{E}$ is only 50 cm away from the target node ($\mathcal{A}$).

Robustness to device compromising. The peculiarities of the EXCHANge protocol, such as high key entropy and low resources request, allow frequent re-generation of the secret key between the two peers. This makes the EXCHANge protocol particularly suitable for scenarios where the adversary $\mathcal{E}$ is allowed to compromise the device and disclose all the secrets in its memory. The adoption of the discussed solution makes device compromising and secrets' disclosure useless from the attacker point of view: in fact the EXCHANge protocol does not need any long-term secret to be stored in memory.

Security. The robustness of the EXCHANge protocol to the RSSI-based source identification attack has been proved by an extensive measurement campaign. Moreover, the discussion provided a lower bound to the number of transmissions (about 569) to generate a secret key of at least 80 bits in very harsh scenario conditions, i.e., $\mathcal{E}$ was placed very close (50 cm) to $\mathcal{A}$, while $\mathcal{B}$ was 6 meters far away from $\mathcal{A}$.

4

Security solutions at the Application Layer for the IoT

The solutions and protocols developed in Section 3 focused on layer-2 security. Thus, they apply between any couple of directly connected devices, having each other direct radio visibility. However, large-scale IoT networks are complex ecosystems, made up of many devices, that can be either many hops away from each other and from the network coordinator. In this case, layer-2 security solutions can still be effectively, but they apply on a hop-by-hop basis, thus resulting in a number of encryption/decryption operations that are directly connected to the number of hops between the source and the destination of the message.

If the two involved devices share a specific session key, instead, the message is not encrypted/decrypted at each hop it traverses, thus providing benefits both in terms of latency of the message and in terms of privacy, because the plain-text message is not revealed to any intermediate device.

As detailed in Section 2.6.3, at the time of this writing a lot of academia and researches are working to provide effective and secure KMP working at the application-layer. A proposal in this context, making use of Implicit Certificates, will be detailed in Section 4.1.

Moreover, the content of this section will provide also some solutions developed to manage the problem of authorization, by leveraging the support of the OAuth 2.0 Authorization Framework. More details will be provided in Section 4.2.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

Finally, the work in the H2020 EU project *symbIoTe* will be described in Section 4.3, with a particular focus on authentication and authorization issues at the application-layer in a federation of heterogeneous large-scale IoT platforms.

4.1 Key Management Protocol with Implicit Certificates for IoT systems

Computational complexity of Public Key Cryptography over sensor nodes is not anymore a blocking concern in modern devices which natively (and efficiently) support Elliptic Curve Cryptography.

Standardization in this field has significantly advanced in the last years. At the application layer, the CoAP has modified the widely employed DTLS so as to fit it into memory/energy constrained devices [56]. At the data link layer, cryptographic extensions have been included in the IEEE 802.15.4 standard, and a new standardization initiative, namely the IEEE 802.15.9 standard [91], is currently addressing the support for KMP messages through IEEE 802.15.4 Information Elements. In the mean time, Public Key Cryptography (PKC) - a fundamental building block in most KMPs - has become viable also on constrained devices. Indeed, this is now the point in time where ECC is not only affordable on today's ever more powerful and (memory) capable constrained devices [121, 128, 129], but it is even cheap and natively introduced in the latest generation of IoT devices [56].

Such a technological evolution is bringing to a point where the primary concern is not anymore the computationally efficient support of ECC, but rather stems in *how to use* such primitives for building *airtime-efficient* authentication and key management protocols. Indeed, most of the proposed PKC-based handshakes described in Section 2.6.3, such as the protocols described in [120], suffer from a significant shortcoming in terms of *number and size of the messages exchanged* [90]. In particular, transmission of long messages containing conventional X.509 certificates [130] yields a sizable *airtime consumption*, whose major consequences are i) a significant latency in the authentication protocol when run over a typical low-rate communication channel, and ii) a significant power consumption, being airtime a major power drain component.

Based on these premises, the contribution of this section is threefold. First, the proposed approach is among the first to concretely integrate and experimentally evaluate

4.1 Key Management Protocol with Implicit Certificates for IoT systems

“implicit” Elliptic Curve Qu-Vanstone (ECQV) certificates [131] within an authentication and key agreement protocol devised for IoT devices and scenarios. While in the former work [90] performance were affected by a software implementation of the ECC primitives, this work shows that the viability of such a technique is greatly improved by the native (e.g., hardware) and efficient support of ECC over modern IIoT devices. Second, the proposed KMP relies on an ordinary and widely established “fixed” ECDH exchange [132], which provides authentication without any explicit signature, as well as ephemeral key derivation (and very fast re-keying, when necessary). This is obtained by exchanging per-session nonces and by securing the exchange using a minimized number of messages (two per each direction, i.e., four in total). Finally, experimental performance results over both single-hop and multi-hop networks show significant improvements in terms of *maximal airtime savings* (up to 86,7%) with respect to a traditional approach relying on an ECDH exchange with public coefficients certified/signed using the ECDSA.

4.1.1 Background: ECQV implicit certificates

For the reader’s convenience, this section briefly reviews the notion of *implicit certificates* along with the details of the ECQV algorithm [131]. Let $\mathcal{G}$ be an Elliptic Curve Group, and let $G \in \mathcal{G}$ be a generator of (prime) order n of the group $\mathcal{G}$. Let U be the bit-string identifying a given user, and let CA be a Certification Authority with Private Key $p_{ca} \in \{0, n\}$, and Public Key $P_{CA} = p_{ca} \cdot G \in \mathcal{G}$. An *implicit* certificate for the user U is a single point $C_U \in \mathcal{G}$ of the group, issued by the CA, which permits a receiver who *knows* the user identity U and the public key P_{CA} of the CA, to *extract* the user’s Public Key $P_U \in \mathcal{G}$ such that $P_U = p_u \cdot G$, with $p_u \in \{0, n\}$ being the user’s private key. Specifically, let $H(\cdot)$ be a cryptographic hash function; then an implicit certificate is issued as follows:

1. the user U generates a random positive integer r , computes an elliptic curve point $R = r \cdot G$, and sends it to the CA;
2. the CA generates a random positive integer k , and returns the implicit certificate as the elliptic point $C_U = R + k \cdot G$, along with the *implicit signature* $\gamma_u = p_{ca} + k \cdot H(C_U, U)$;
3. the user generates the private key $p_u = \gamma_u + r \cdot H(C_U, U)$ and the relevant public key $P_U = p_u \cdot G$.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

Version [1 B]	Version of the X.509 certificate (set to 3)
Serial Number [8 B]	Unique identifier of the certificate
Signature [1 B]	Signature of the certificate (set to NULL, not useful for Implicit Certificates)
Validity [10 B]	Validity of the certificate in the time domain
Issuer [8 B]	The CA that generates the certificate
Subject ID [8 B]	The user that asks for a certificate
Public Key Algorithm ID [1 B]	Cryptographic algorithm associated to the public key stored in the certificate (set to ECQV)
Public Key	Public Key (set to ECQV implicit certificate, i.e., $P_{i,u}$)
Signature Algorithm ID [1 B]	Algorithm used to generate the signature (set to NULL, not useful for Implicit Certificates)

Figure 4.1: X.509-compliant implicit certificate format, main fields, from [1].

Any party can trivially compute U 's public key by knowing only its identity U , its implicit certificate C_U , and the CA's public key P_{CA} , as $P_U = P_{CA} + H(C_U, U) \cdot C_U$. Indeed,

$$P_{CA} + H(C_U, U)C_U = \{p_{ca} + (r+k)H(C_U, U)\}G = p_u G = P_U.$$

The security of such a construction was formally proven in [133], and an X.509-compliant implicit certificate format using in total 78 bytes was presented in [1] (see Fig. 4.1).

4.1.2 Key Management Protocol

The proposed Key Management Protocol (KMP) relies on the widely accepted ECDH handshake, but minimizes airtime consumption by exchanging the public ECDH coefficients in *implicit format*, i.e. by using ECQV implicit certificates. The design challenge is therefore to comply with the “fixed” nature of the resulting handshake (ephemeral ECDH coefficients would mandate an explicit digital signature). This is addressed by the exchange and handling of fresh nonces per each session, via two supplementary *authentication* messages following the initial ECDH exchange, which guarantee peer authentication and (fresh) key derivation.

4.1 Key Management Protocol with Implicit Certificates for IoT systems

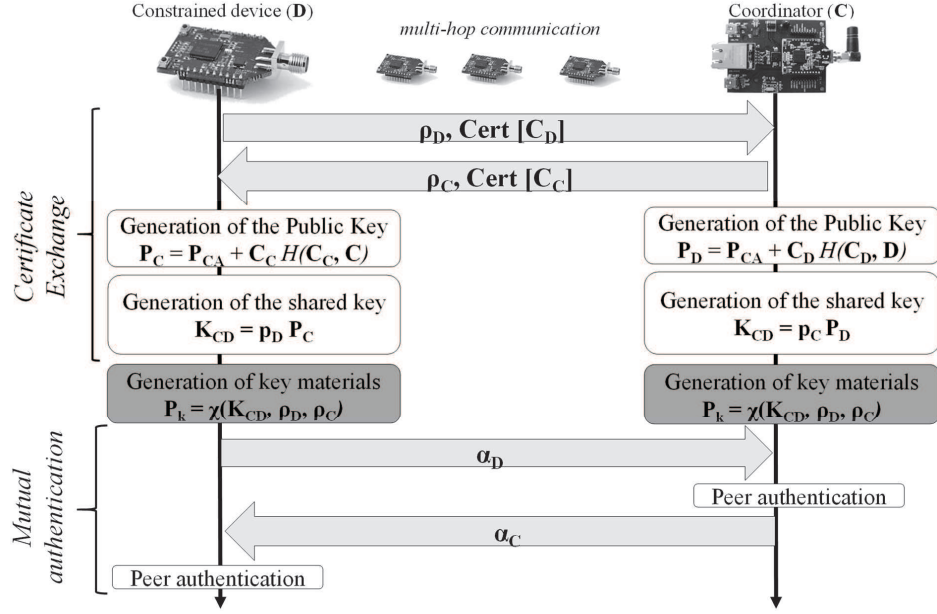


Figure 4.2: Key negotiation protocol.

In details, the proposed protocol, non restrictively illustrated in Fig. 4.2 for the case of a constrained device D which mutually authenticates and negotiates a session key with a coordinator device C , comprises two phases. Without loss of generality, the KMP assumes that all the constrained devices, including the network coordinator, trust the same Certification Authority (CA), and possess the following cryptographic material: own public key, own private key, own implicit certificate in the X.509 format (as described in section 4.1.1), and CA's public key.

In the first phase, each peer X (i.e., C or D) sends to the other party a message comprising i) a nonce ρ_X , and ii) an implicit ECQV certificate C_X . The receiver is thus able to:

1. extract the Public Key $P_X = P_{CA} + C_X \cdot H(C_X, X)$ from the implicit certificate C_X , as described in section 4.1.1, and
2. compute the (same and fixed) pre-master session key K_{CD} using an ordinary ECDH. For instance, once received C_D and after having extracted P_D , the network coordinator C can use its private key p_c to compute $K_{CD} = p_c P_D = p_c p_d G$; the same result is obtained on the other side by the device D which computes $K_{CD} = p_d P_C = p_c p_d G$.

Note that (as indeed expected by a “fixed” ECDH exchange) subsequent handshakes among the same pair of devices would yield the same key K_{CD} . Moreover, note that any attacker could *replay* the message containing the implicit certificate so as to impersonate

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

one of the two devices.

To address both of these issues, the proposed Key Management Protocol relies on fresh nonces in each “new” exchange. Nonces serve for two complementary purposes:

1. Authentication. To authenticate the exchange, peers D and C compute an *authentication tag*

$$\alpha_D = \Gamma[K_{CD}, (C_D, D, C_C, C, \rho_D, \rho_C)]$$

$$\alpha_C = \Gamma[K_{CD}, (C_C, C, C_D, D, \rho_C, \rho_D)]$$

where the $\Gamma[k, s]$ operator refers to a generic symmetric authentication algorithm (e.g., an HMAC) working on the bit stream s by using key k . Note that the two tags differ, as they use the same information but in a different order [132].

2. Key Derivation. Finally, each peer computes the actual session key P_K (or multiple keys, to distinguish encryption key from integrity key as customarily done in mainstream security protocols) via an ordinary/standard Key Derivation Function (KDF) χ , as: $P_K = \chi(K_{CD}, \rho_D, \rho_C)$ - details in section 4.1.4.

It is worth noting that the proposed KMP intrinsically supports fast opportunistic re-keying. In fact, based on the fixed ECDH mechanism, the key negotiation algorithm always produces the same shared secret for a specific device pairs. Therefore, the heavy cryptographic operations required by both ECQV and ECDH algorithms can be avoided in a subsequent handshake: a device can retrieve from a local cache the previously computed pre-master key and simply compute a new session key using the new exchanged nonces.

4.1.3 Security considerations

The proposed approach relies on building blocks whose security is widely established, namely ECDH [132] and ECQV [133]. Note that these building blocks remain *independent* in the proposed construction, opposed to being composed, and secret parameters for both schemes are neither exposed to external entities nor used in a way different from their standard usage.

In this way, it is possible to avoid potential issues that may emerge when mixing blocks whose conjuncted adoption is not universally guaranteed ([134], for instance, shows problems related to the composition of implicit certificates and ECDSA technique).

4.1 Key Management Protocol with Implicit Certificates for IoT systems

Moreover, ECQV implicit certificates, binding a public key to its owner in a trusted way, make the proposed strategy robust against MITM attacks. Furthermore, the mutual authentication scheme implemented in the second part of the protocol protects the entire approach against replay attacks, and explicitly binds the exchanged cryptographic quantities to the involved peer identities using per-session nonces. It is worth noting that the two authentication messages closely mimic the operation of the *Finished* message in the TLS protocol, and therefore inherit the relevant security properties assessed for the TLS protocol [108]. Indeed, each authentication tag is computed by including *all* the information exchanged in the first two messages (plus the peer identities) and hence protects the entire exchange from MITM modifications.

Finally, the designed protocol does not specifically influence (i.e. neither positively or negatively) resilience against physical attacks such as tampering, fault, and side-channel attacks, resilience which is mandated to a careful technical implementation and choice of the involved Elliptic curves. For instance, standard software/hardware-based techniques can be used to mitigate tampering attacks and prevent the physical access to security parameters stored within the device. To prevent fault attacks which force the victim device to perform calculations on weak elliptic curves in order to leak the secret key, it is necessary to carefully select the considered ECC curve: the one adopted in the implementation (see Section 4.1.4 for details) satisfies this requirement. And in terms of side-channel attacks (at least for what concerns side timing-channels), the ECC hardware implementation adopted in the prototype employs the *Montgomery ladder* [132] algorithm (see Section 4.1.4) and thus guarantees that the time needed to perform ECC point multiplication is independent from side-channel information and does not leak secret key information.

4.1.4 Implementation

The proposed KMP scheme has been implemented in the OpenWSN protocol stack described in Section 1.4.2.1, installed on the OpenMote-CC2538 constrained platform whose features has been highlighted in Section 1.4.1.3, and released as open source at the site <http://telematics.poliba.it/iot-ecc>, under the Berkeley Software Distribution (BSD) license. Some implementation aspects and design choices are listed below.

1. The KMP has been developed at the application layer and integrated in CoAP. The

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

KMP instance running on a given node is handled as a CoAP resource, identified with the Uniform Resource Identifier (URI) “*coap://[ip-addr]:[port]/kmp*”, where *[ip-addr]*, *[port]*, and *kmp* are the IPv6 address of the node, the port number (default value 5683), and the KMP resource, respectively.

2. At the link-level, the IIoT technology supports very small packets (e.g., Maximum Transmission Unit equal to 127 bytes). KMP messages carrying peer certificates thus need to be fragmented. Then, the proposed KMP used the *block-wise* transfer mode [135].

3. CoAP provides a reliable transfer by acknowledging packets after reception. To reduce bandwidth and energy consumption, and reduce delay (to a greater extent in multi-hop scenarios) the KMP employed a *piggybacking* mechanism, where fragments of each peer’s certificate are transmitted together with the ACK message referring to the previously received packet.

4. The OpenMote-CC2538 cryptoprocessor provides hardware accelerated atomic cryptographic ECC operations, which have been properly assembled and further optimized via software configuration. The well-known curve *secp160r1* provided by NIST [136] is used for handling ECC operations. At the time of this writing, this is the first contribution that integrates elliptic cryptography on OpenMote-CC2538 boards in the OpenWSN protocol stack. From the security perspective, *secp160r1* is able to guarantee the minimum acceptable security level (e.g., equal to 80) envisaged for an elliptic curve [137], while ensuring the lowest bandwidth requirements. Moreover, such a curve is considered resilient against fault attacks [138]. Furthermore, the OpenMote-CC2538 cryptoprocessor also implements the *Montgomery ladder* algorithm, used to make ECC operations resilient against side-channel attacks. The authentication tag is also generated by a hardware accelerated execution of the HMAC-SHA256 algorithm.

5. The session key P_K is currently computed by using the *MGF1* function, specified by IEEE P1363a [139], which provides a good compromise among simplicity, security, and code footprint. The code can be trivially extended to support better key derivation functions, such as the provably-secure KDFs recommended in the IETF RFC 5869.

6. For simplicity, the KMP does not use ECC point compression techniques to further reduce the size of the implicit certificate. Due to fragmentation of the application payload, the first two logical messages are always sent through 2 link-layer packets.

7. The implemented KMP based on implicit certificates requires 12.240 kBytes of ROM

4.1 Key Management Protocol with Implicit Certificates for IoT systems

and 644 Bytes of RAM, whereas the KMP implementation based on X.509 explicit certificates used in the next section for benchmarking purposes uses 15.192 kBytes of ROM and 877 Bytes of RAM. It therefore follows that implicit certificates also bring about advantages in terms of memory footprint.

4.1.5 Experimental performance assessment

The airtime consumption requested by the proposed KMP has been experimentally evaluated in a network with a variable number of devices organized in a chain topology. A benchmark approach that uses the same KMP procedure and adopts explicit X.509 certificates encoded through the standard Privacy Enhanced Mail (PEM) format and the binary Distinguished Encoding Rules (DER) format, as well as digitally signed via the ECDSA algorithm, has been taken into account for the comparison. The link layer, based on the IEEE 802.15.4e technology, has been configured with a slotframe of 101 slots, each lasting 10 ms. During a slotframe, each node has only two active slots: one for the management of the control traffic (i.e., routing messages, beacon frames, and keep-alive packets), the other one for exchanging KMP-related packets. As a result, a 2% duty-cycle for each node is ensured.

Figs. 4.3 - 4.5 show the impact that cryptographic and communication functionalities have on each single atomic operation. These results have been obtained by testing the execution of the KMP in a single hop network. As expected, implicit certificates can significantly reduce the handshake time. This performance gain is mainly obtained thanks to two different aspects. First, the 78 bytes implicit certificates can be sent within just 2 link-layer packets, opposed to the 13 packets needed for the 725 bytes of the explicit X.509 certificate in the PEM format, or the 9 packets required to send the 495 bytes of the explicit X.509 certificate in the DER format. Second, cryptographic operations of ECQV require a lower computational load with respect to that needed to compute an ECDSA signature.

Fig. 4.6 reports (with 95% confidence intervals) the time required to complete the KMP protocol as a function of the number of hops that separates the involved nodes. These experimental results clearly demonstrate that the usage of implicit certificates always ensures the maximal airtime saving, with performance gains over explicit X.509 PEM certificates ranging from 77,1% to 86,7%, and from 50,9% to 84,7% with respect to explicit X.509 certificate in the DER format.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

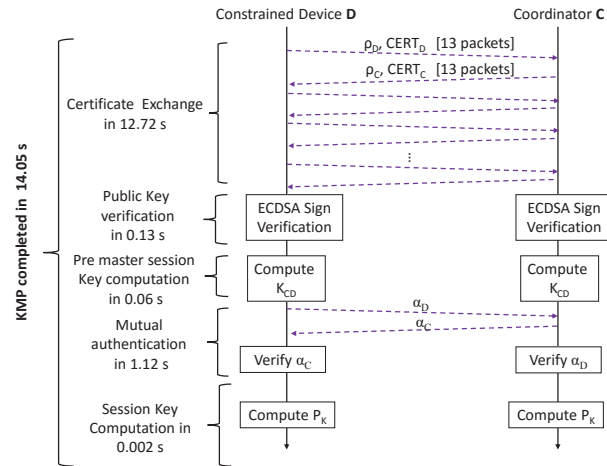


Figure 4.3: Computational load of atomic operations with explicit X.509 certificates, PEM format.

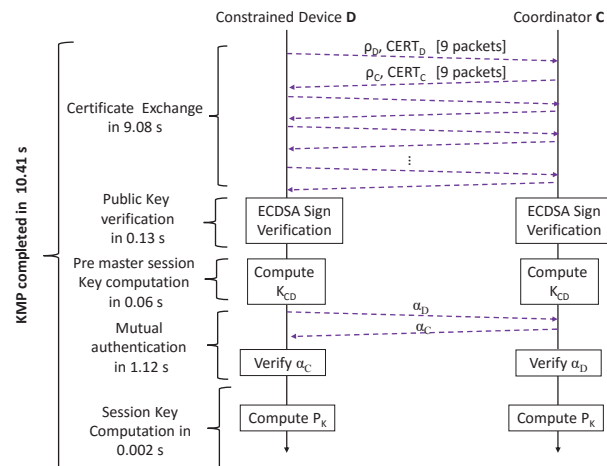


Figure 4.4: Computational load of atomic operations with explicit X.509 certificates, DER format.

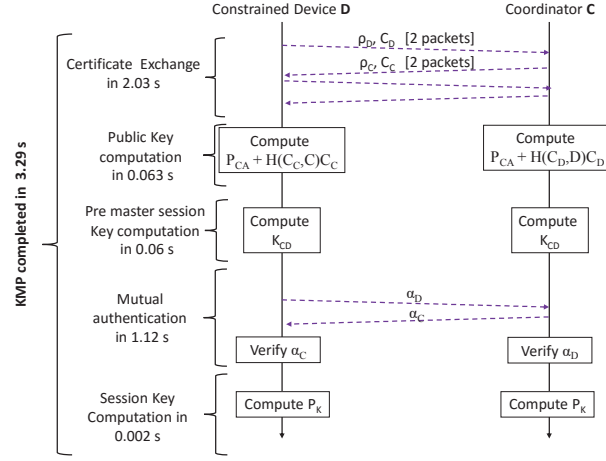


Figure 4.5: Computational load of atomic operations with implicit X.509 certificates.

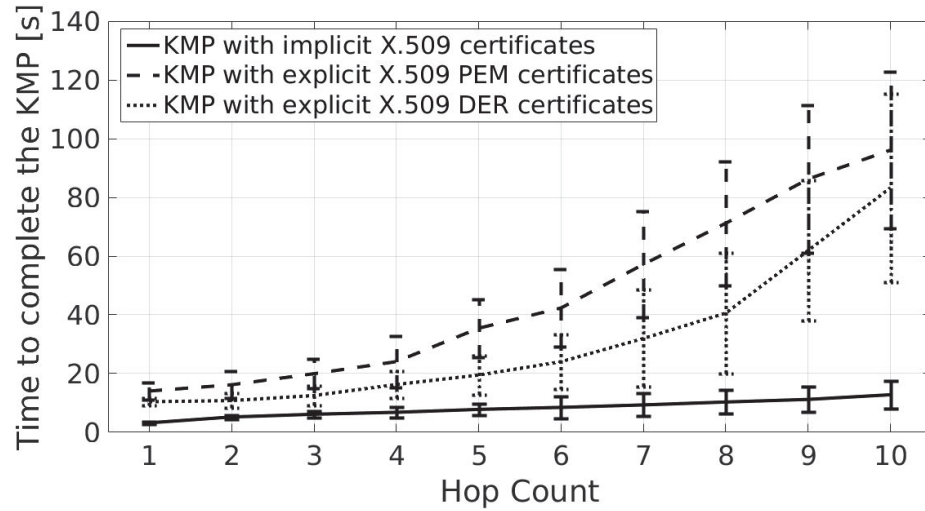


Figure 4.6: Airtime consumption.

4.2 Authorization in IoT systems

All the previous works, including the KMP described in Section 4.1, focused on security protocols and mechanisms applied directly on constrained devices.

When dealing with authorization issues, instead, applying security functionalities directly on the constrained device does not represent a smart solution.

Given the target scenario, in fact, applying security countermeasures on the gateway of the IoT network can relieve smart objects from managing complex rules and authorization mechanisms, providing benefits both in terms of latencies and energy consumption.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

While many *turnkey* solutions are already hitting the market for both data confidentiality and key agreement [91] [56], the discussion on authentication and authorization services, instead, is still open to novel and effective solutions. In this context, the target scenario is simple, but challenging at the same time: IoT networks expose restricted resources outside (i.e., in Internet); third-party applications may be interested to reach these resources; and access requests must be authorized by resource owners.

In Internet, this problem was smartly solved by the OAuth 2.0 authorization framework [95]. Some papers taking into account OAuth 2.0 for managing authorization issues in the IoT have been explained in detail in Section 2.6.3. Nevertheless, a fully standard-compliant scheme that allows the straightforward integration of OAuth 2.0 and accepted communication protocols for the IoT is still missing.

In line with these premises, this section presents the novel OAuth-IoT framework, which offers resource access control by leveraging and properly harmonizing existing open-standards, while taking care of the limited capabilities of constrained devices. In OAuth-IoT, IoT resources are discovered and exposed through well-known mechanisms standardized by the IETF [140]. A Resource Server stores details on available resources, provides an interface between the IoT network and the Internet, verifies authorization permissions of third-party applications interested to access to exposed resources, and implements a caching mechanism for efficiently serving multiple requests with limited freshness requirements. OAuth-IoT also integrates an Authorization Server that authenticates the resource owner and authorizes a third-party application to access to one of the aforementioned IoT resource. While the entire process is completely based on the OAuth 2.0 specification, authorization grants are handled through Proof-of-Possession (PoP) tokens [141]. Furthermore, in order to show the effectiveness and benefits of the proposed framework, an experimental testbed has been developed and made publicly available at <http://telematics.poliba.it/oauth-iot>.

To completely understand the scope and functionalities of the OAuth-IoT testbed, a preliminary discussion about the OAuth 2.0 and the most important token formats introduced in the IETF context is necessary.

4.2.1 The OAuth 2.0 authorization framework

The OAuth authorization framework, now available in the version OAuth 2.0, was designed in order to allow a third-party application to access to restricted resources under

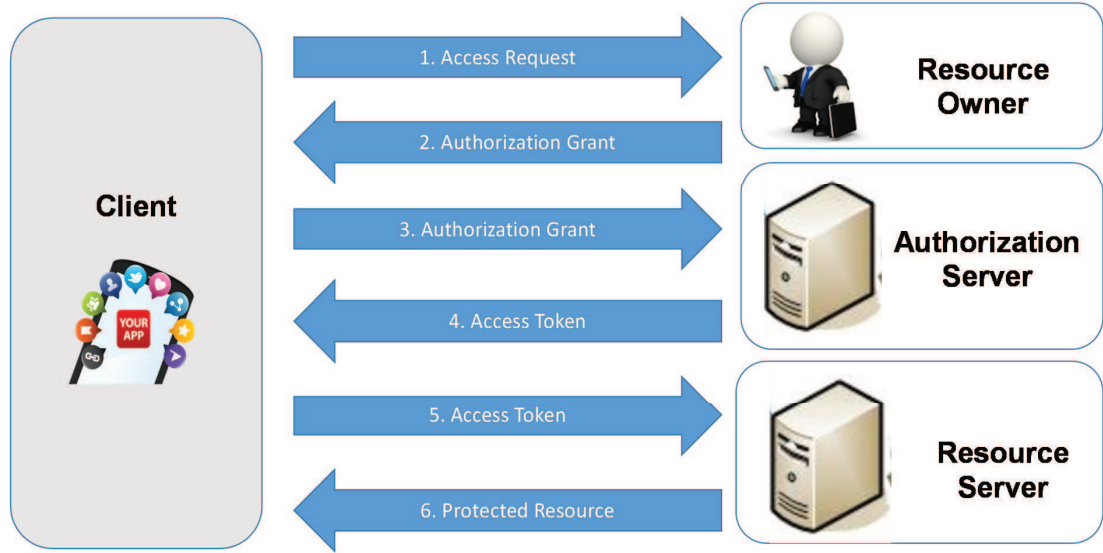


Figure 4.7: Protocol flow for the OAuth 2.0 access control framework.

the control of a user, without requiring the sharing of user's credentials [95]. OAuth 2.0 defines four actors: a third-party application willing to access to protected resources (i.e., the *client*), the *Resource Owner* able to grant or deny access to a protected resource, the *Resource Server* that exposes protected resources, and the *Authorization Server* that controls the authorization process. As described in Figure 4.7, the OAuth 2.0 authorization framework involves six interactions: (1) the client contacts the Resource Owner of the resource; (2) the Resource Owner (RO) grants the access to the client by sending an *authorization code*; (3) the client delivers the received authorization code to the Authorization Server; (4) the Authorization Server verifies the authorization code and releases a token containing the details of the consent provided to the client (time limit, scope, and so on); (5) the client forwards the token to the Resource Server; (6) the Resource Server checks the validity of the received token and, in affirmative case, it provides the protected resource. Note that all the aforementioned messages are exchanged over a TLS-secure channel.

OAuth 2.0 specifies two kind of tokens: access tokens and refresh tokens. The former is a time-limited string encoding the authorization issued by the client, and it is basically used to access the resource. The latter, instead, is used to obtain new access tokens and, therefore, it has longer validity.

However, the definition of the token structure is out of scope of the standard.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

4.2.2 IETF token formats

Concrete proposals for tokens come from other activities in the IETF context.

For instance, *Bearer Tokens* are defined in [142] as simple containers of information. The majority of OAuth 2.0 implementations adopts this solution, even if no embedded security mechanisms are provided. In this case, token confidentiality is delegated to the mandatory TLS protocol.

The *JSON Web Token (JWT)* is a compact way for carrying the access grants [143]. It leverages the standard JSON format for storing and assess common claims (including time validity, issuer, owner and revocation time) and authentication fields. There is also the possibility to extend the token format with unregistered private claims, useful for specific purposes.

Finally, the *Proof-of-Possession (PoP) tokens* have been defined in [141] and can be used in scenarios demanding additional security protection. In fact, the main idea behind PoP tokens is that a client must demonstrate possession of cryptographic keying material when making requests for accessing protected resources. Both symmetric and asymmetric cryptography can be used in this case.

4.2.3 The proposed OAuth-IoT framework

The OAuth-IoT framework aims at offering access control functionalities for the IoT, by properly leveraging the potentials and the functionalities of accepted open standards. Figure 4.8 reports the target architecture. It embraces the following components:

- **Internet of Things network:** it integrates many constrained devices able to sense the surrounding environment, acquire data (such as temperature, humidity, luminosity, and acceleration) and deliver them to a sink node, the *network coordinator*, through a low-power and short-range wireless communication technology. The sink node is attached to the *Resource Server*, that acts as a gateway device and offers many functionalities listed below.
- **Client:** in line with the OAuth 2.0 authorization framework, it is a third-party application willing to reach resources belonging to an IoT network, on behalf of the resource owner.

- **Resource Server:** it covers same functionalities as in the OAuth 2.0 authorization framework. First of all, it exposes outside resources available in the IoT network (i.e., in Internet). Therefore, it represents the contact points of any external client willing to reach IoT resources. Furthermore, it implements other key functionalities related to security (e.g., establishment of confidential communication links, client authentication, and access control) and system performance (e.g., caching of data and freshness control).
- **Authentication Server:** it manages authorization mechanisms, as detailed by the OAuth 2.0 authorization framework.

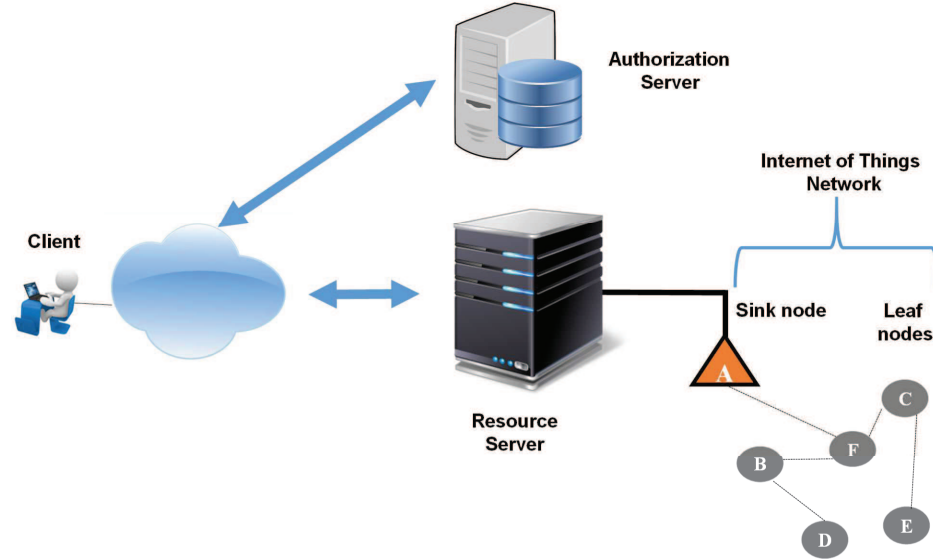


Figure 4.8: Reference architecture.

In OAuth-IoT, the client can access remote resources through authorization primitives already standardized for the OAuth 2.0 framework. Therefore, it is necessary to design an effective interface between OAuth 2.0 and the IETF protocol stack described in Section 1.3.5, while integrating a concrete implementation for tokens and proper mechanisms able to reduce energy consumption within the IoT domain. To this end, OAuth-IoT fully leverages (without introducing any modifications) the current reference open standards and orchestrates the following main features:

1. Dynamic configuration of the *Resource Directory*, stored in the Resource Server;

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

2. OAuth-driven access control to IoT resource;
3. User authentication through PoP tokens;
4. Energy efficient management of user queries.

The main technical details related to the OAuth-IoT framework are reported below.

4.2.3.1 Resource management at the Resource Server

The Resource Server represents the first point of contact between the OAuth 2.0 authorization framework and the IoT. In fact, it is in charge of storing information related to available resources, as well as exposing them to the rest of the architecture through OAuth 2.0 authorization primitives. To reach this goal, it hosts three different data structures, that are *Routing Table*, *Resource Directory*, and *Data Table*. They are populated thanks to the continuous interaction between the sink node of the IoT network and the Resource Server. The *Routing Table* contains the identifier of all nodes currently active in the IoT network, along with information about the path that is used to communicate with each of them. The *Resource Directory* stores all the details related to the resources available in the IoT network (note that each constrained node may expose one or more resources). Finally, the *Data Table* stores the last updated data provided by each node, along with the associated meta-data, i.e., the node that provides the resource, the identifier of the resource on this node and the timestamp associated with the data.

Within the IoT network, according to the RPL protocol described in detail in Section 1.3.5.5 the sink node periodically sends DIO messages for discovering new nodes. A new node that joins the network answers with a DAO message. All the information carried by the DAO message are delivered by the sink node to the Resource Server and then stored within the *Routing Table*.

Once a new record is added to the *Routing Table*, the Resource Server initiates the resource discovery procedure by polling the identified constrained device at the path *./well-known/core*. The corresponding answer is properly processed for populating the *Resource Directory*.

Note that the client is able to access resources exposed through the *Resource Directory*. Accordingly, the interaction between the Resource Server and the sink node represents a key feature provided by the developed OAuth-IoT framework.

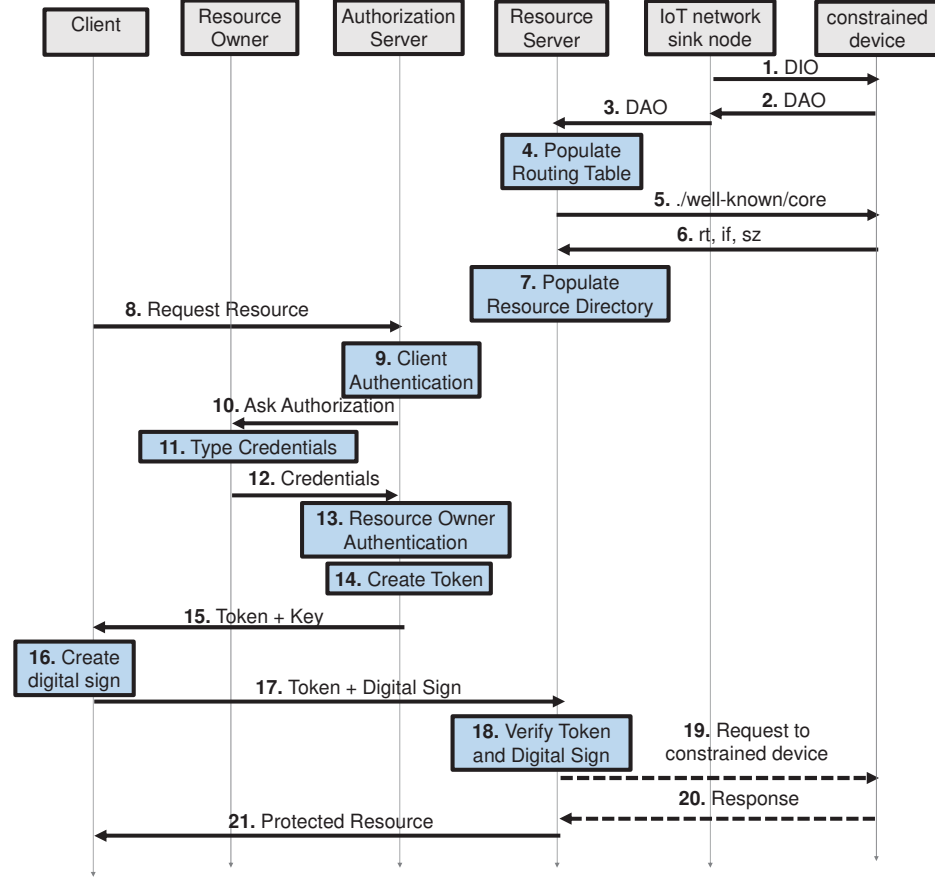


Figure 4.9: Message Sequence Chart for a client accessing to a resource on the RS.

When a new data is acquired from the IoT network (i.e., after the processing of an external request), the Resource Server updates the *Data Table*. Specifically, each record of that table stores identification data about the resource (i.e., the Resource Type attribute previously received with the Core Link Format message), the IPv6 address of the node exposing the resource, the timestamp (i.e., the time instant at which the data has been generated), and the value of the data itself. In other words, the *Data Table* acts as a cache and can be used to generate answers to external requests, thus limiting the overall energy consumption in the IoT network.

4.2.3.2 Sequence diagram for the resource access procedure

Figure 4.9 shows the sequence diagram for the resource access procedure implemented in the developed OAuth-IoT framework.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

At the top-right of the figure, it is possible to observe the interaction between the Resource Server and the IoT network, needed to populate the data structures reported in the previous sub-section. Specifically, the sink node delivers the DIO messages and waits for the related DAO messages produced by all the leaf nodes in the network. Thanks to these information, the sink node creates the *Routing Table*. Then, the *Resource Server* polls each node in the *Routing Table* on the *./well-known/core* path to have the list of resources produced by the specific node. Thanks to these information, the Resource Server creates the *Resource Directory*, and makes it available to external applications.

On the left-side of the figure there is the client that would access to a resource available in the IoT network. To this end, it starts the authorization procedure based on the OAuth 2.0 work flow. At the beginning, it sends a request to the Authorization Server that, in turns, asks the authorization to the resource owner. The resource owner (e.g, the user that is using the application) types username and password, authorizing the access. Then, the Authorization Server generates the token and forwards it to the client. Note that in this step, the Authorization Server also delivers to the application a secret (i.e., the key), that will be used to prove that the application is the owner of the token, i.e. the entity for which this token has been generated (more details will follow in Section 4.2.3.3). The application creates the digital sign of the token and forwards it, along with the token and the content of the original request, to the Resource Server. The Resource Server processes the token (according to the procedure described in Section 4.2.3.3) and generates the answer (according to the procedure described in Section 4.2.3.4). It is worth noting that the RS can quickly response to the application if the requested resource is already in the database and meets the conditions requested by the client (i.e., freshness). If this is not the case, the HTTP request is mapped to a CoAP request and forwarded to the node that provides it. For this reason, the messages 19 and 20 in Figure 4.9 are optional and they have been illustrated through dotted lines.

4.2.3.3 Details on tokens

OAuth-IoT uses symmetric PoP tokens [141]. The considered Internet draft assumes that the Authorization Server generates a unique symmetric key, k , every time a new

client requests an access token. This key is stored within a dedicated database, shared also with the Resource Server.

Now, with reference to the sequence diagram reported in Figure 4.9, the Authorization Server sends, at the step 15, the token T and the symmetric key k . The token stores a confirmation claim CNF , computed as in the following:

$$CNF = H(K), \quad (4.1)$$

where $H(\cdot)$ refers to a hash function.

The application must demonstrate that it is the real owner of the token. To this end, it uses the received key for calculating a digital sign of the token itself:

$$SIGN_T = E(H(T), K), \quad (4.2)$$

where $E(\cdot)$ and $H(\cdot)$ refer to a symmetric encryption algorithm and a hash function, respectively. Therefore, at the step 17 in Figure 4.9, the application delivers to the Resource Server the token and the calculated digital sign.

To validate the token, the Resource Server retrieves the key from the database shared with the Authorization Server (AS) and verifies that both the confirmation claim stored in the token and the digital sign sent by the client are correct (i.e., they correspond to the same values locally calculated by the Resource Server). In the affirmative case, the Resource Server provides the requested resource to the application, by means of the procedure reported in Section 4.2.3.4.

4.2.3.4 Operations performed by the Resource Server

The set of operations executed by the Resource Server are illustrated in Figure 4.10.

First of all, when the Resource Server receives a new request, it extracts the token, the identifier of the constrained node exposing the resource, the requested resource, and a data freshness parameter. The procedure described in Section 4.2.3.3 is immediately executed for verifying the validity of the received token. Thus, the user is authorized to access to the resource if and only if the token is not expired and the user is able to demonstrate the possession of the symmetric key provided by the Authorization Server. Otherwise, the Resource Server returns the well-known *403 forbidden* error code and rejects the request.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

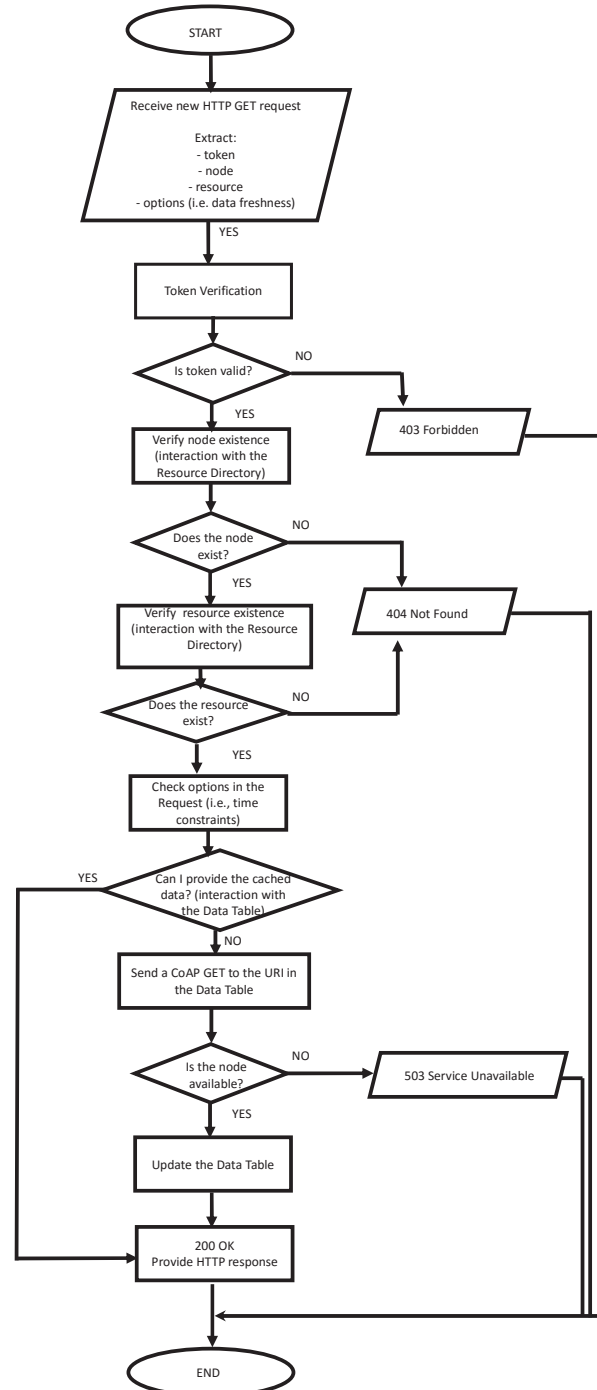


Figure 4.10: Flow diagram of the operations executed by the RS when a new HTTP request arrives from the Internet.

If the authorization procedure ends successfully, the Resource Server checks the availability of the requested resource in the *Resource Directory*. In the case the resource is not available, the Resource Server returns the well-known *404 not found* error code and closes the session. Otherwise, the rest of the operations reported in Figure 4.10 are executed. At this moment, the Resource Server checks the presence of the requested data within the *Data Table*. If a record is found and the associated timestamp satisfies the freshness requirement reported in the request, the Resource Server immediately delivers to the client the cached value (i.e., without contacting the constrained node in the IoT network that exposes the requested resource). If not, the Resource Server generates a GET request using CoAP and sends it into the IoT network. Now, in the case the node is not reachable, the *503 service unavailable* error code is generated and the session is closed. On the contrary, the contacted constrained node answers with the requested data that will be processed and stored within the *Data Table* of the Resource Server (first) and then pushed back to the client (then).

4.2.4 OAuth-IoT Testbed at work

An experimental testbed has been developed to practically demonstrate the main functionalities of the conceived OAuth-IoT framework. It is available at:

<http://telematics.poliba.it/oauth-iot>

The IoT network hosts three OpenMote constrained devices [42]: one sink node and two leaf nodes exposing resources (i.e., temperature, humidity, light, and acceleration). They run the OpenWSN stack [44], described in Section 1.4.2.1.

The OpenWSN protocol stack has been properly extended with specific CoAP applications, dedicated to provide available resources to the RS; they are */light*, */temp*, */humid* and */accel*, for light, temperature, humidity and acceleration, respectively.

Both Authorization Server and Resource Server components are implemented on the Pandaboard ES platform. AS and RS processes are implemented by using the open-source python framework Django¹ and the open-source database PostgreSQL².

¹<https://www.djangoproject.com/>

²<http://www.postgresql.org/>

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

Moreover, the AES encryption algorithm and the Secure Hash Algorithm (SHA) hashing algorithm, used for the management of PoP tokens, are implemented through the open-source PyCrypto library¹.

Resources can be requested by using a web-based application, directly connected with a web server running on the Resource Server. It has been realized by using the open-source framework JQuery² and it is able to manage multiple AJAX calls.

Figure 4.11 shows the GUI of the OAuth-IoT testbed with the three tasks controlled by the client. The client can select the constrained node (i.e., **ed1a** and **ed8a**), the resource (i.e., **/light** for the brightness, **/temp** for the temperature, **/humid** for the humidity, and **/accel** for the acceleration), and the desired freshness (i.e., 10 seconds). The request is then processed by the Authorization Server, that initiates the authentication procedure. A recognized user has **username = test** and **password = oauth-iot**. Finally, the client can proceed to effectively send the request to the Resource Server. The answer is reported at the bottom of the web-page.

The screenshot displays the OAuth-IoT testbed GUI, organized into three main sections: Client, Authorization Server, and Request Resource. The Client section includes radio buttons for node selection (ed1a is selected), a lightning bolt icon for resource selection (/temp is selected), and a clock icon for freshness selection (10 is selected). A black Submit button is located below these options. The Authorization Server section features a user icon with the text 'test' and a lock icon with a masked password '.....'. A black Login button is positioned below the login fields. The Request Resource section mirrors the Client section with the same options (ed1a, /temp, 10) and a black Continue button. At the bottom of the interface, a black status bar displays the text 'ed1a : /temp = 25.40°C : 0'.

Figure 4.11: GUI of the OAuth-IoT testbed.

To provide a further insight, some preliminary performance tests have been carried out for demonstrating the impact of the freshness on the amount of time needed to

¹<https://pypi.python.org/pypi/pycrypto>

²<https://jquery.com/>

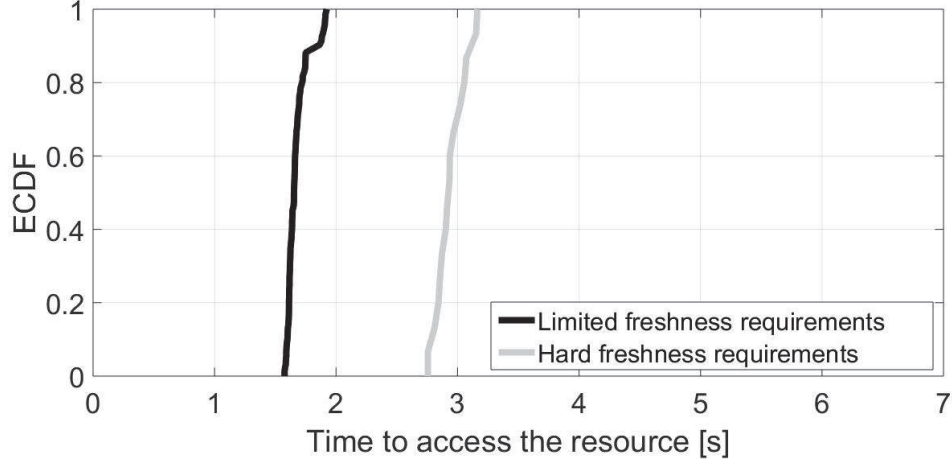


Figure 4.12: Latencies measurements.

successfully reach the requested resource. The empirical cumulative distribution function (ECDF) of the measured latencies is reported in Figure 4.12. The black curve refers to the latency measured for a request with a limited freshness requirement (e.g., the answer is always gathered from the *Data Table*). The grey curve, instead, refers to the latency measured for a request with a hard freshness requirement (e.g., the answer is always generated by the constrained node in the IoT network). From the figure it clearly emerges that the caching functionality offered through the *Data Table* significantly reduces latencies (the answer is always provided in less than 2 s). An updated value, instead, can be reached in less than 3.5 s. As a final consideration, it is important to note that overall system response time is strictly influenced by the configuration of the IoT network. In the developed testbed, the IoT network uses a simplified TSCH schedule made up of three slots lasting $10ms$, as described in [33]. Different choices, that may depend on the network load and the service to be provided, can modify these performances indexes.

4.3 The symbIoTe H2020 project

Authentication and authorization issues have been taken into careful consideration also within the major European projects in the IoT context. Between these, *symbIoTe* is a cornerstone position.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

In fact, while IoT introduced the possibility to connect electronic things from everyday life to the Internet, making them ubiquitously available, the new research trend is following the development of advanced services, based on a trusted federation among heterogeneous IoT platforms.

However, a new set of security problems (including authentication and authorization) emerges. The aim of this section is to describe the main facets of the preliminary security architecture envisaged in the context of the *symbIoTe* (namely, symbiosis of smart objects across IoT environments) project, recently launched by European Commission under the Horizon 2020 EU program.

4.3.1 *symbIoTe* high-level description

In a world of smart networked devices, wearables, sensors and actuators, transparent and secure access to available resources across various IoT domains is crucial to satisfy the needs of an increasingly connected society. However, the current IoT ecosystem is fragmented: many vertical solutions run nowadays which, from one hand, integrates connected objects within local environments using dedicated implementations and, on the other hand, they connect smart spaces with a back-end cloud hosting often dedicated proprietary software components [2].

The *symbIoTe* project aims at facilitating the creation and management of hierarchical, adaptive and dynamic IoT environments, while devising an interoperability framework across existing and future IoT platforms. The interoperability framework enables a seamless networking and rapid cross-platform application development.

Figure 4.13 sketches an environment of the future *symbIoTe* ecosystem which is built around a hierarchical IoT stack and spans over different IoT platforms.

symbIoTe assumes that various IoT devices, namely Smart Devices, are connected to IoT gateways within Smart Spaces, representing physical environments with deployed *things*. The platforms provide IoT services locally in Smart Spaces and share the available local resources (e.g., connectivity, computing and storage). Smart Spaces and local platform services are connected to same services running in the cloud (e.g. resource discovery and management, data analytics). In addition to their local representation, the physical things deployed in the Smart Space are also mapped to their virtual representations within the cloud, and exposed as IoT services to third parties, e.g., mobile or web applications [2].

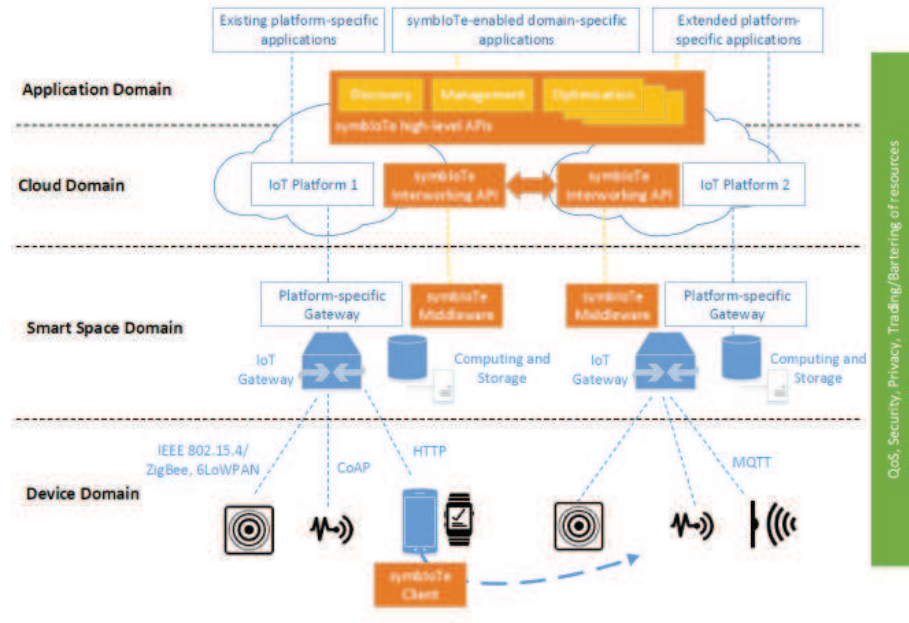


Figure 4.13: Sketch of the *symbIoTe* architecture [2].

The *symbIoTe* architecture spans over the following four layered domains depicted in Figure 4.13:

- *Application Domain*: it enables the creation and management of cross-platform virtual environments, to allow a unified view on various platforms and exposed resources.
- *Cloud Domain*: it enables platforms interoperability and the creation of platform federations, so that they can securely inter-operate, collaborate and share resources;
- *Smart Space Domain*: it enables dynamic discovery and configuration of resources within local smart environments, even those already connected to different platforms;
- *Smart Device Domain*: it relates to heterogeneous IoT devices representing things in smart spaces; IoT devices can be discovered in visited Smart Spaces and use their resources in a controlled way (device roaming).

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

4.3.2 Attribute-Based Access Control in federated IoT platforms

The aim of the following discussion is to describe the main facets of the preliminary security architecture envisaged in the context of the *symbIoTe* project. At the time of this writing, the developed solution already offers distributed and decoupled mechanisms for the provisioning of authentication and authorization services in complex scenarios embracing many, heterogeneous, and federated IoT platforms. Specifically, they leverage Attribute Based Access Control and token-based authorization techniques. Thus, this section will provide a step-by-step description of security functionalities in three different target scenarios and will suggest, at the end, some promising technical implementations that will be taken into account in the hereafter of the *symbIoTe* project.

4.3.2.1 Architecture

The reference architecture considered in this contribution is depicted in Figure 4.14. It integrates many independent IoT platforms exposing heterogeneous resources. Each IoT platform (thus, each available resource) is registered with a trusted mediator (i.e., the interoperability framework's *core*) which offers advanced mechanisms for enabling platform interoperability and distributed resource access. Moreover, there are applications willing to access the available resources.

To maximize interoperability among platforms the security framework has to deal with different scenarios: applications can be registered to only the trusted mediator, to only one IoT platform, or two (or more) IoT platforms federated with the mediator entity. Therefore, the following target scenarios can be identified:

- **Scenario #1:** an application is registered with an IoT platform and it would like to access resources exposed by the IoT platform where it is registered to. This is the case of a typical closed system, where applications, services and resources are controlled by the same administrator, without the need to interface with other platforms.
- **Scenario #2:** an application is registered with the trusted mediator and it would like to access resources exposed by a federated IoT platform. This is the case of a third-party application developer, that implements special applications to access services and resources exposed by a given IoT platform controlled by a different administrator.

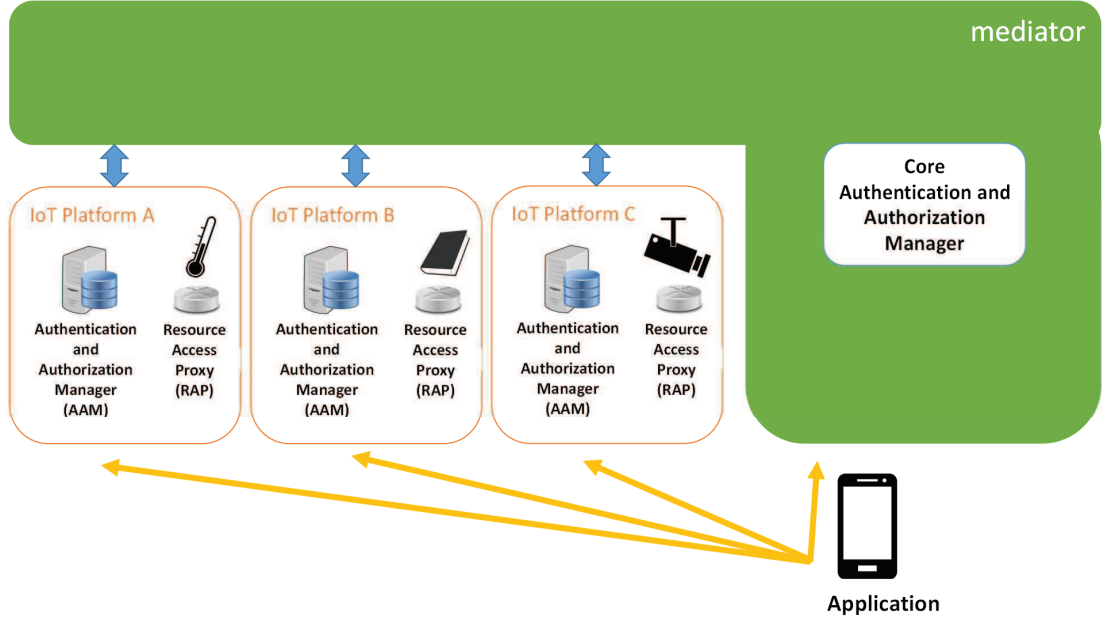


Figure 4.14: System Architecture.

- **Scenario #3:** an application is registered with one or more IoT platforms federated with the mediator and it would like to access resources exposed elsewhere in the considered architecture. This is the case of a current sensor in a private smart home. To access the data, the system could require an application to register both with Smart-Home and the Service Provider IoT platforms. This scenario is referred as the *multi-domain access rights composition* paradigm.

4.3.2.2 Main security rationale

The resource access is handled through the Attribute Based Access Control (ABAC) logic. ABAC is a well-known technique for dealing with access control in distributed environments, which is able to protect sensitive data, applications or services from unauthorized operations by means of efficient, simple and flexible access rules. It is based on *attributes* and *access policy* concepts.

An attribute encodes a specific property, role or permission assigned to an application. Attributes are stored within a digital object, namely a *token*, that certifies the authenticity of both the issuer (i.e., a dedicated component of mediator or IoT platform) and the owner (i.e., the application), additionally to its time validity. Both

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

symmetric or asymmetric cryptography techniques can be used to ensure authenticity and integrity of those tokens.

An access policy, instead, enables a fine-grained access control mechanism. In fact, it describes the combination of attributes needed to obtain the access to a given resource. For each resource, a dedicated access policy can be defined. Then, an application in possession of tokens storing a set of attributes matching the aforementioned access policy can successfully obtain the access to the resource. Otherwise, its access request will be denied.

It emerges that the token represents a key element in the resource access mechanism. From the security perspective, it is generated during the authentication procedure and inspected and validated during the authorization procedure. The solution described in this contribution natively offers the decoupling between authentication and authorization processes. This means that authentication and authorization involve different components and are independently executed at different times. An application uses the authentication procedure to authenticate itself within a given domain (like the trusted mediator or an IoT platform federated with the mediator). In case of a successful authentication it obtains a set of tokens storing its own attributes. Then, the collected attributes can be used during the authorization procedure to obtain access to resources. Since an application should not perform the whole authentication process for each resource access, the designed approach allows also for enhanced flexibility and scalability benefits for the whole system.

Note that when an application or component registered in a given IoT platform or in the mediator would like to access resources exposed elsewhere, it could be possible that the attributes that are assigned to it are not valid also in the new domain. Therefore, an *Attributes Mapping Function* is needed to manage the translation between attributes in different platforms.

Thanks to the described functionalities, at the same time the interoperability framework works on top and extends the existing security architecture of a given IoT platform, providing procedures for allowing secure communications with foreign IoT platforms and third-party applications.

4.3.2.3 Component description

To practically implement the aforementioned security rationale in each target scenario, the following logical components are introduced:

Platform Authentication and Authorization Manager (AAM): With reference to **Scenario #1** and **Scenario #3**, it handles the authentication procedure for applications registered with the IoT platform. Therefore, it releases *home* tokens storing attributes that describe properties, roles and/or permissions assigned to the application within the platform where it is registered to. It also manages a Token Revocation List (TRL) storing the list of tokens that have been revoked before their expiration. For this reason it may be contacted by any component in the architecture during the *check revocation procedure*.

With reference to **Scenario #2** and **Scenario #3**, the Platform AAM is in charge of (i) verifying the validity and the possible revocation of tokens generated by the AAM component of the mediator or the AAM component of another IoT platform, (ii) performing the attributes mapping functionality and (iii) releasing a new set of tokens, namely *foreign tokens*, usable in the local IoT platform.

Core AAM: With reference to **Scenario #2**, it handles the authentication procedure for applications registered with the mediator. Therefore, it releases *core* tokens storing attributes that describe properties, roles and/or permissions assigned to the application at the mediator side. Moreover, similarly to the Platform AAM, it manages the TRL and can be contacted by any component in the architecture during the *check revocation procedure*.

Resource Access Proxy (RAP): With reference to **Scenario #1**, **Scenario #2** and **Scenario #3**, it holds the URI for obtaining all the resources available in the specific IoT platform, along with the access control policy associated to each of them. Therefore, it receives all the requests for accessing these resources along with the tokens containing the attributes of the requester. Finally, it enforces the access control by checking if the provided attributes satisfies the policy associated with the resource. In the positive case it provides access to the resource itself, otherwise the access is denied.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

Note that each AAM component manages authentication and authorization functionalities only for users and applications that are registered within its reference domain. In this sense, third-party applications that are not registered with any reference IoT platforms registers within the core AAM. This design choice provides enhanced flexibility to the interoperability framework, because each AAM does not need to store data related to all possible components in the systems. Instead, being responsible only for local components, each AAM stores only a limited amount of data. When foreign components tries to access to local services, instead, the reference AAM can interact with the foreign AAM to obtain the required information about the requesting application.

4.3.2.4 Sequence Diagrams

The reference sequence diagrams for the scenarios introduced at the beginning of this discussion will be provided in what follows. They suppose a previous registration phase, in which each component in the system receives an asymmetric key pair (private/public keys) and a public-key certificate in a trusted way. The public key of the application is also included in the token, to guarantee authenticity of the token itself and to provide the cryptographic material to be used and verified in the challenge-response procedure. Finally, end-to-end security in the communication between any component in the interoperability framework is guaranteed thanks to the mandatory adoption of the well-known TLS protocol [107].

The sequence diagram describing the resource access in **Scenario #1** is depicted in Figure 4.15. It is composed by two main steps:

Step 1: Home authentication. At the beginning, the application performs the login in it's home platform by contacting the home AAM and receiving home tokens.

Step 2: Resource access authorization. The application contacts the RAP and delivers the home tokens retrieved in the previous step. The RAP initiates the challenge-response mechanism to verify that the application is the real owner of the tokens. In the case the challenge-response mechanism is successfully completed, the RAP verifies that tokens are valid and that they have not been revoked by contacting its reference platform AAM component. Then it checks the provided attributes against the access policy associated with the requested resource:

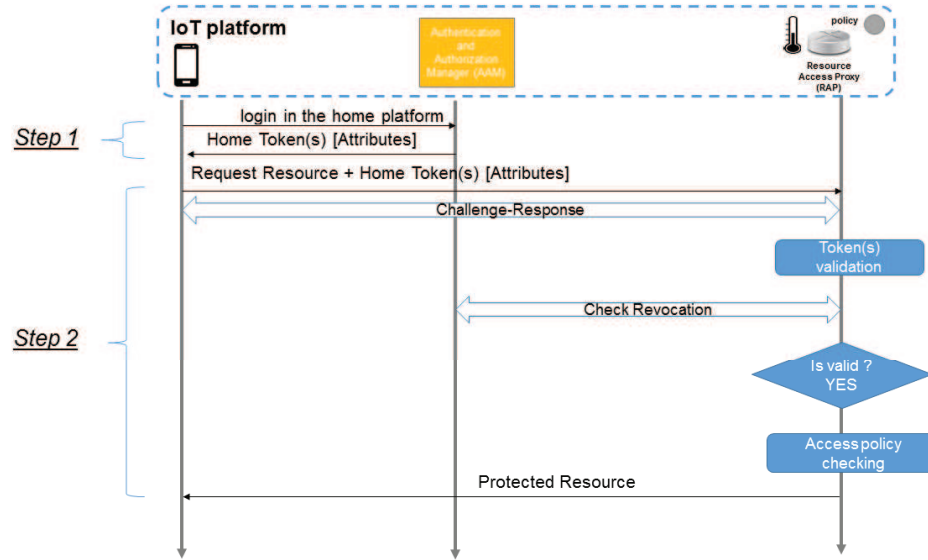


Figure 4.15: An application registered in a given IoT platform wants to access resources produced within its home IoT platform.

if the attributes supplied by the applications are enough to satisfy the access policy associated with the resource (according to the ABAC logic) the RAP grants the access to the resource. Otherwise access is denied.

The sequence diagram describing the resource access in **Scenario #2** is depicted in Figure 4.16. It is composed by three main steps:

Step 1: Core authentication. At the beginning, the application performs the login with the mediator by contacting the core AAM and receiving core tokens.

Step 2: Foreign authentication. The application forwards core tokens to the AAM component of the foreign platform. The AAM component of the foreign platform initiates the challenge-response mechanism to verify that the application is the real owner of the tokens, thus preventing both replay and impersonation attacks. In the case the challenge-response mechanism is successfully completed, the AAM component of the foreign platform validates the tokens, verifies that they have not been revoked by contacting the AAM component of the mediator and performs the attribute mapping function. Then, it generates a new set of foreign tokens and sends them to the application.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

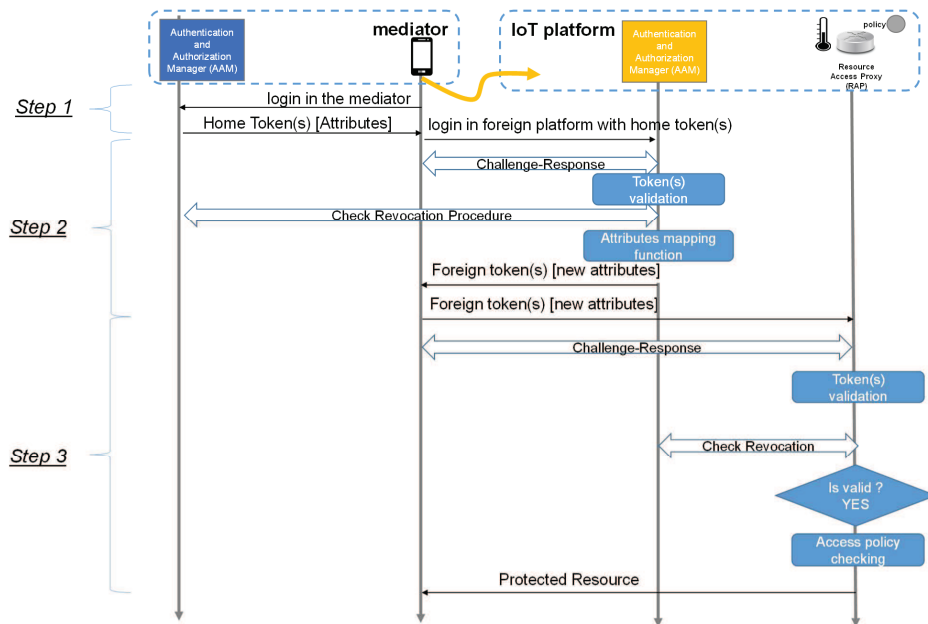


Figure 4.16: An application registered in the mediator space wants to access resources in an IoT platform.

Step 3: Resource access authorization. The application contacts the RAP and delivers it the foreign tokens retrieved in the previous step. The RAP initiates the challenge-response mechanism to verify that the application is the real owner of the tokens. In the case the challenge-response mechanism is successfully completed, the RAP verifies that tokens are valid and that they have not been revoked by contacting its reference platform AAM component. Then it checks the provided attributes against the access policy associated to the requested resource: if the attributes supplied by the applications are enough to satisfy the access policy associated to the resource (according to the ABAC logic) the RAP grants the access to the resource. Otherwise the access is denied.

The sequence diagrams describing the resource access in **Scenario #3**, referring to as *multi-domain access rights composition*, are depicted in Figure 4.17. Without loss of generality, it is assumed that the application is registered in platforms *IoT_A* and *IoT_B* and would like to gain access to a resource available in the platform *IoT_C*. Also in this case, the procedure has three main steps:

Step 1: Home authentications. At the beginning the application performs the login in the IoT platforms where it is registered to (i.e., *IoT_A* and *IoT_B*). To this end it contacts the AAM component of each platforms for retrieving home tokens.

Step 2: Foreign authentication. The application combines the home tokens and forwards them to the AAM component of the foreign platform *IoT_C*. The AAM component of the foreign platform initiates the challenge-response mechanism to verify that the application is the real owner of the tokens, thus preventing both replay and impersonation attacks. In the case the challenge-response mechanism is successfully completed, the AAM component of the foreign platform validates the tokens, verifies that they have not been revoked by contacting AAM components of the home platforms (i.e., *IoT_A* and *IoT_B*) and performs the attribute mapping function. Then, it generates a new set of foreign tokens and sends them to the application.

Step 3: Resource access authorization. The application contacts the RAP and delivers the foreign tokens retrieved at the previous step. The RAP initiates the challenge-response mechanism to verify that the application is the real owner of the tokens. In the case the challenge-response mechanism is successfully completed, the RAP verifies that the tokens are valid and that they have not been revoked by contacting its reference platform AAM component. Then it checks the provided attributes against the access policy associated with the requested resource: if the attributes supplied by the applications are sufficient to satisfy the access policy associated to the resource (according to the ABAC logic) the RAP grants access to the resource. Otherwise, the access is denied.

4.3.2.5 Planned implementation

The implementation of the security architecture described in Section 4.3.2.1 requires the selection of a suitable token format. Without loss of generality a token is a digital object used as a container for security-related information. It serves for authentication and/or authorization purposes and generally appears as a list of elements. Each element contains an assertion that further specifies properties assigned to the owner of the token.

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

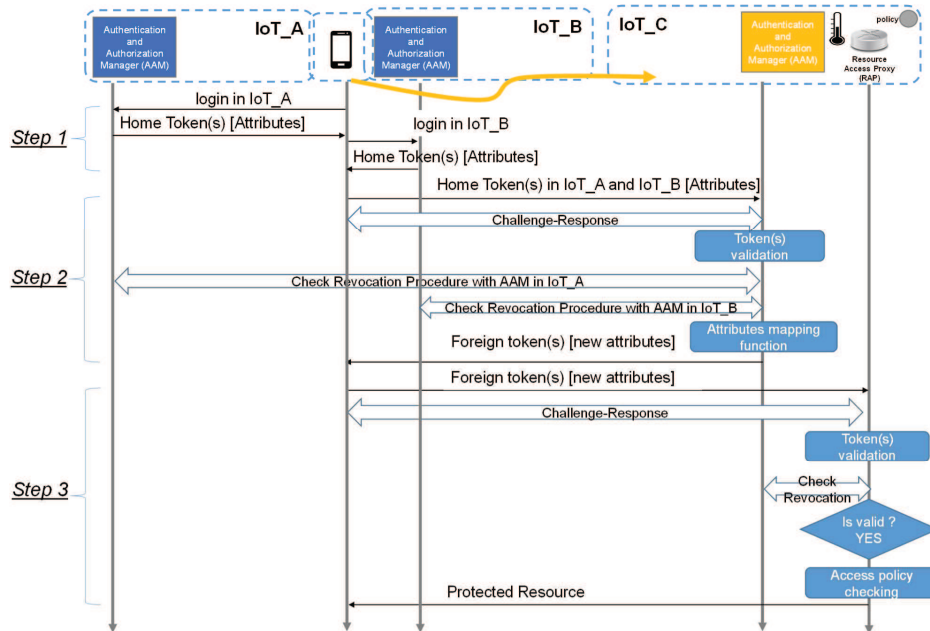


Figure 4.17: Multi-Domain Access Rights Composition.

Each token must contain an explicit expiration date, indicating the date until the token can be considered valid. Moreover, the token also contains at the end an element that certifies its authenticity and integrity. Depending on the chosen solution, validating a token could require different procedures.

The following content describes two promising technical solutions, candidates for the implementation of tokens in the proposed approach. During implementation of the framework itself one or a combination of the following technologies will be used.

Macaroons: Macaroons are a recent kind of authorization credential developed by Google [144]. As bearer credentials, they serve a similar purpose as cookies in World Wide Web, but they are more flexible and provide better security. Macaroons are based on a construction that uses nested, chained MACs (e.g., Hash-keyed Message Authentication Code (HMAC)) in a way that is highly efficient, easy to deploy and widely applicable. They allow authority delegation between bearers with attenuation and contextual confinement. Each field embedded within macaroons structure, i.e. the caveat, restricts both the macaroons' authority and the context in which it may be used (e.g. by limiting the permitted actions and requiring the bearer to connect from a certain IP address and to present additional evidence such as a third-party

signature). Macaroon caveats are plain-text readable. Macaroons also contain a list of *AND* conditions. Its bearer is authorized to perform an operation *AS LONG AS* *condition1, condition2, ..., conditionN* hold true. The main (root) macaroon which allows for everything gets successively attenuated with those conditions. Each of them is signed with an HMAC function.

By considering the reference architecture shown in Figure 4.8, three possible tokens can be introduced: *root macaroons*, *platform macaroons*, and *application macaroons*. The mediator creates a root macaroon by calculating the HMAC function of a random nonce and its secret key. Note that the output of the HMAC function must be shared among AAMs of federated platforms for verifying the authenticity of tokens received during the resource access procedure. Starting from the root macaroon the mediator also generates platforms macaroons. The platform macaroons are signed by the HMAC function with the key being the previously calculated HMAC value. Then, each platform can autonomously generate application macaroons by following the same process.

An application, after obtaining the macaroon from the AAM component of its platform, may further attenuate it and therefore authorize others to perform actions in its name. For example a Smart Home System mints its owner a full access token. The owner can go on vacation and want the neighbor next door to be able to operate the windows and doors but not the garage. The owner can do so by attenuating his/her own token and handing it to the neighbor.

JSON Web Tokens: JWT is an open industry standard widely used in today's Internet to deal with authentication and authorization issues [143]. It contains a set of *claims*. A claim is a specific certified statements related both to the token itself or to the entity that is using it. Typically, these claims are encoded in the JSON format, thus easily allowing system interoperability.

A claim is identified with a specific name: it is possible to distinguish between *Registered Claim Names*, that are names defined and standardized in the reference document and *Private Claim Names*, that represent extensions that a developer could choose for his/her own system.

The cryptographic force of the JWT resides in the sign field, stored at the end of the token. It can be generated through symmetric or asymmetric cryptography techniques and allows to verify the authenticity of the token, i.e. generation by a trusted entity, as

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

Table 4.1: Registered Claim Names of the JWT and related description.

Registered Claim Name	Description
iss	It uniquely identifies the entity that issued the token
sub	It uniquely identifies the entity for which this token has been released (it is a key field when a token needs to be used also for authentication purposes)
exp	It indicates the expiration time, after which this token should not be used and processed by any entity in the system
nbf	It identifies the time in which this token becomes effectively valid and can be processed by any entity in the system
iat	It uniquely identifies the time in which this token has been created
jti	It is the unique identifier of the token

well as integrity, in the sense that no one could modify its content without invalidating it.

Each JWT contains a header that provides information about the type of the token and the algorithm used to build the sign of the token. It contains also a body, encoding a set of claims for this token, and finally - a sign containing the cryptographic validation of the token and generated as stated in the header.

Table 4.1 depicts Registered Claim Names that can be carried in the body of the JWT.

JWT fits perfectly within the reference architecture described in Section 4.3.2.1. From a cryptographic perspective, the only requirement for its adoption is the deployment of a public-key infrastructure, which issues a private/public key pair to each entity in the system.

AAMs uses their private keys to generate and sign tokens. Any entity in the system that receives a token could easily verify its authenticity by gathering the public key of the issuer of the token (specified in the token itself).

nonce	
access all	
time constraints for platform-token	
sign_aam_id = $S(Pv_{root}, \text{iot-a})$	Header: Alg: RSA/ECDSA Typ: JWT
time constraints for user-token	jti = id
list of attributes $[A_1, A_N]$	iss = iot-a
user certificate (user id)	sign_iss = $S(Pv_{root}, \text{iot-a})$
sign_user_id = $S(Pv_{\text{iot-a}}, \text{user_id})$	iat [value]
sign_user = $HMAC_{\text{platform-token}}(\text{token})$	nbf [value]
	exp [value]
	sub = user certificate
	att = list of attributes
	token_sign = $S(Pv_{\text{iot-a}}, \text{token})$

(a)

(b)

Figure 4.18: Details of the content of (a) Macaroons and (b) JWT tokens for the designed architecture.

Important features such as the support for an expiration date are integrated by JWT thanks to the definition of the *exp* claim.

Also, each token can be easily associated to a given entity in the system through the *sub* claim. More in detail, the public key of the owner of the token can be embedded in this claim. This can be used in the challenge-response procedure described in Section 4.3.2.1 to prove the possession of the respective private key and verify that the application using the token is effectively the entity for which the token has been generated. This procedure avoids replay attacks.

Finally, the JWT can be easily extended to support the carrying of attributes associated to the ABAC logic, thanks to the possibility to integrate customized Private Claims.

Comparison: To conclude, the comparison between the user macaroon and the user JWT is reported in Fig. 4.18.

The application macaroon is showed in Figure 4.18(a). It has a hierarchical structure. The nonce is used as a unique identifier of the token. The second, third and fourth caveats are related to the AAM of the platform in which the application is registered

4. SECURITY SOLUTIONS AT THE APPLICATION LAYER FOR THE IOT

to. In particular, the ID of the AAM is signed through the private key of the mediator entity (PV_{root}). The remaining lines are dedicated to the application. They contain the list of attributes assigned to the application, its public key certificate and, finally, the sign on the user ID by the AAM that issued the token, through its private key (PV_{iot-a}). The last line is needed to assure that the AAM is the unique component able to sign the token. Finally, the last caveat is the chained HMAC of all the caveats in the token, performed starting from the output of the HMAC function of the root macaroon.

The application JWT is showed in Figure 4.18(b). Also in this case it is possible to identify the part related to the issuer of the token, certified through the sign with the private key (PV_{root}), and the part related to the owner of the token. The private claim *att* is introduced to encode the information about the list of attributes possessed by the application. Finally, the sign of the whole token is performed through the private key of the issuer (PV_{iot-a}) without the need of a symmetric shared secret.

Note that both token types have a limited time validity. After the expiration of that date, the token must be renewed through a new authentication procedure.

The evaluation of pros and cons of macaroons tokens and JWTs as well as their suitability for the proposed scenario is left for future work.

5

Time-based wireless localization techniques

5.1 Time-based localization logic

Many IoT applications require positioning capabilities within a bounded area such as within a city, along main vehicle or rail links, or within specific venues such as an exhibition, entertainment or health-care facilities. Usually, localization capabilities for these services are provided through the well-known Global Positioning System (GPS) system. Positioning reliability in these areas can be improved by using cellular, Wi-Fi or IoT signals as well as or in combination with GPS to provide accurate positioning. Moreover, in certain situations GPS signals cannot be available, due to particular environmental conditions or the presence of GPS jamming signals. Furthermore, for some class of devices the GPS positioning system could be too expensive, both to provision and to apply, due to computational and energy constraints. Leveraging these ideas, this section proposes a two-stage localization technique that can be used to improve positioning of Wi-Fi, Cellular and IoT devices if combined with legacy GPS approaches, or to replace the GPS technique itself in case it is not available, to provide advanced combination of localization techniques.

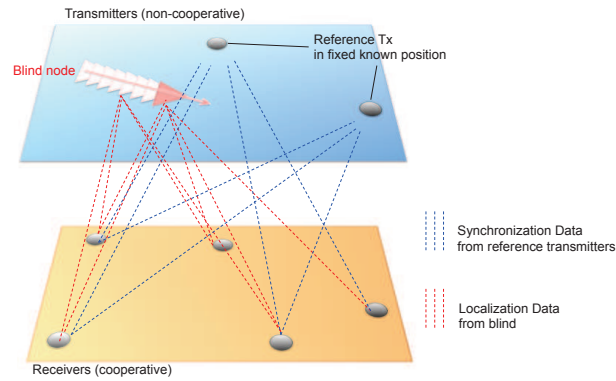
5.2 System Overview

This work tackles the problem of passively identifying the trajectory of a moving radio source from Time-of-Arrival (TOA) measurements collected by a set of coop-

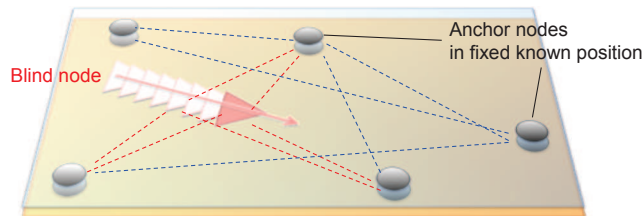
5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

erating receivers. To maximize applicability and minimize cost of deployment, the system-level requirements are kept down at a minimum. Receiver node clocks are left un-synchronized, and, equivalently, TOA measurements are synchronized in post-processing, leveraging measurements gathered from other (non-cooperative) reference transmitters. No assumption is made about transmission times, thus enabling the opportunistic exploitation of non-cooperative transmitters.

The general reference scenario is depicted in Fig. 5.1(a).



(a) General scenario with independent reference transmitters (non-cooperative) and receivers (cooperative).



(b) Particularized scenario: each anchor node serves both as receiver and reference transmitter.

Figure 5.1: Reference Scenario.

The reference wireless system includes a blind node transmitting a sequence of packets, while moving linearly at constant velocity. In parallel, a set of fixed reference transmitters in known positions are also transmitting.

Both types of packets (from the moving blind node and from the fixed reference transmitters) are sent at arbitrary (unknown) transmission times, i.e., transmissions are completely asynchronous. A set of receivers in fixed known positions are configured

to overhear and timestamp each received packet, and these data are shared with a central entity in charge of the computation. The problem is to determine the initial *position* and the *velocity* vector of the blind node from the timestamps recorded during a given measurement interval.

A batch processing approach is considered, where all measurements collected within the observation window are jointly processed in a single run to estimate *ex post* the trajectory parameters (initial position and velocity), with no prior knowledge about the initial node state. This problem can be considered as the generalization to a moving node of the classical localization problem for a static node, hence we refer to it with the term “generalized localization”, or alternatively “tracing” [145].

The tracing problem should be distinguished from the “tracking” problem, where the *current* instantaneous position of the blind node is iteratively updated upon arrival of each new measurement starting from an initial state that is known at least approximately.

The considered system is completely asynchronous and is characterized by the following features:

- (a) Receiving nodes are asynchronous: each node clock is affected by unknown *time offset* and *frequency offset* (clock skew).
- (b) Packet transmissions are asynchronous: no explicit control is exerted on the transmission times.
- (c) Transmission times are unknown: the tracing process relies exclusively on *reception* timestamps.
- (d) The blind node moves with unknown (constant) velocity.

It is possible to highlight that, owing to (b) and (c), transmitters are not required to cooperate explicitly with the tracing process. This is a key aspect of the considered system, as it allows to exploit *opportunistically* packets transmitted for communication, not for localization purposes. A minimum level of cooperation is assumed solely between the (fixed) receivers for the purpose of sharing the collected measurements. Moreover, note that in this scenario the blind node acts purely as transmitter, hence it does not participate to — and might be unaware of — the localization/tracing process.

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

In the general scenario depicted in Fig. 5.1(a) reference transmitters and receivers are regarded as independent entities. For several real-world applications, particularly those involving WiFi deployments and IoT, it makes sense to consider a particularization of this model with a single set of anchor nodes in (known) fixed positions, each anchor serving both as (reference) transmitter and receiver. In other words, reference transmitters and receivers are 1:1 coupled, as depicted in Fig. 5.1(b). Motivated by the practical relevance of this scenario, and by the advantage of a certain notational simplification in this case, the following discussion focuses the presentation of the proposed method and of the numerical results to the particular scenario of Fig. 5.1(b), with the understanding that all proposed algorithms can be generalized straightforwardly to the more general case of independent transmitters depicted in Fig. 5.1(a).

This scenario was first introduced in a recent letter [145] with reference to the ideal case of complete (lossless) reception, wherein every transmitted packet is received by every receiver in the system. This follow-up discussion considers the more realistic scenario of incomplete (lossy) reception.

This work formalizes the tracing/localization problem within the system model outlined above and present a possible resolution strategy. Along the way it provides different variants of Least-Squares (LS) estimators based on TOA and Time-Difference-of-Arrival (TDOA). This study proves the equivalence of the TOA and TDOA instances for the problem at hand — a result that strengthens and generalizes previous results about TOA and TDOA equivalence. Numerical results from Monte Carlo simulations are provided to explore the impact of various system-level aspects and parameters. For the special case of null blind velocity (localization problem), the proposed method is compared against an alternative approach based on Differential Time-Difference-of-Arrival (DTDOA) that was published in a previous work [3]. Finally, results are validated on experimental data from a real-world WiFi testbed.

5.3 Previous and related works

Careful analysis of the literature revealed that there is no solution in the literature that can be applied to the considered scenario, since all previously proposed methods fail to meet one or more of the system requirements (a)-(d) laid down in Sec. 5.2.

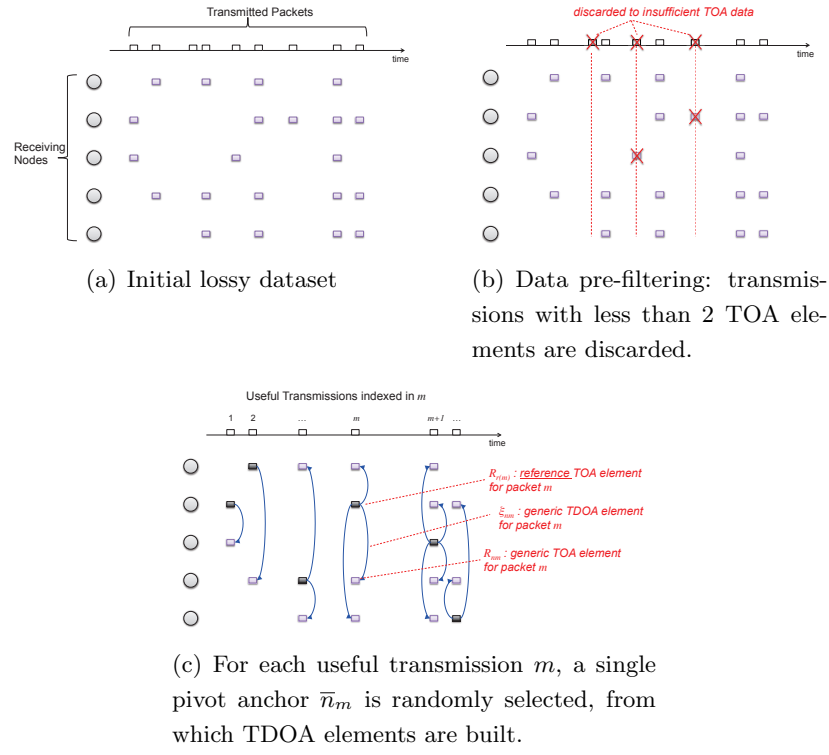


Figure 5.2: Graphical representation of dataset pre-processing. Stage (c) is needed only for TDOA.

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

The vast majority of previous work on time-based localization in asynchronous networks, including e.g. [146, 147, 148, 149, 150, 151, 152, 153, 154, 155] rely on the knowledge or tight control of transmission times (in addition to reception times) and ultimately require some form of cooperation by the blind node. This applies also to the solutions based on two-way ranging. For this reason, these methods can not be used to implement fully opportunistic localization on top of legacy wireless systems without transmitter cooperation, and therefore can not be directly compared to our work.

In [156, 157] the authors have considered node clock with zero frequency offset, effectively assuming “quasi-synchronous” scenario instead of “fully asynchronous” (following the taxonomy proposed in [146]) and static blind nodes, thus missing requirements (a) and (d). The only previous work considering all three conditions (a)-(c) are [3, 158] and more recently also [159, 160]. However, the resolution algorithms proposed in these papers were designed specifically for a static node (missing condition (d)) and cannot be extended easily to the case of a moving node. More specifically, the solutions proposed in [3, 158, 160] are range-based, i.e., they rely on the intermediate estimation of distances (or distance differences, or pseudo-ranges) between the blind node and the anchors, and therefore do not generalize straightforwardly to the case of moving nodes where blind-anchor distances vary at each measurement time. Similarly, the algorithms proposed in [159] rely on linear and/or semidefinite programming approximations of the core ML estimation problem that are tailored to the static scenario.

The equivalence between TOA and TDOA formulations was noted in some previous work, limited to the particular case of a static node. However, the notion of “equivalence” as well as the interpretation of “TOA formulation” differs across these papers. To illustrate the differences, consider two distinct estimators for the same variable of interest, and let us introduce the following terminology:

- Weak equivalence $\stackrel{\text{def}}{=}$ the two estimators lead to different solutions having the same variance.
- Strong equivalence $\stackrel{\text{def}}{=}$ the two estimators lead to exactly the same solution.

Furthermore, for non-linear estimators that are resolved by numeric methods, we define:

- Stronger equivalence $\stackrel{\text{def}}{=}$ the two estimators lead to the same solution and do so across the same search path.

Most previous work had shown only weak equivalence results between TOA and TDOA formulations (for static nodes) based on the equivalence of Cramer-Rao bounds. The only explicit strong equivalence result appears in [161, Theorem 2], but limited to linearized (approximated) versions of the TOA and TDOA estimator. Instead, no previous work has delved into the different facets of “equivalence” between TOA and TDOA (and S-TOA) for the general non-linear case, neither for static nor for dynamic case.

5.4 The system model

5.4.1 Notation and assumptions

The reference scenario is depicted in Fig. 5.1(b). Consider a system with $N + 1$ nodes indexed in $n = 0, \dots, N$. Index $n = 0$ is reserved for the (moving) blind node, whose trajectory parameters are to be estimated, whereas the remaining N (anchor) nodes are placed in fixed known positions.

The system assumes a minimum of five anchors, i.e. $N \geq 5$. During the observation interval, the blind node transmits M (data) packets, while every anchor transmits B (beacon) packets. The notation $G \stackrel{\text{def}}{=} BN$ indicates the total number of beacon packets transmitter by the anchors. Each packet may be overheard and timestamped by all other anchor nodes (except the sender in case of beacon packets), and all the collected measurements are shared with a central entity in charge of the computation. In an ideal system with zero packet loss, every packet is timestamped by all potential receivers, resulting in a complete dataset of $MN + G(N - 1)$ measurements. However, in real systems occasional packet loss cannot be avoided (due e.g. to interference or collisions) and therefore must be taken into account into the system model. To illustrate, Fig. 5.2(a) graphically sketch an example of lossy dataset, where each transmitted packet (top row) is received by only a subset of possible receivers.

Since the transmission time is unknown, a generic packet produces useful information for the localization/tracing problem only if it is received (and timestamped) by two or more anchor nodes. For this reason, we preliminarily discard the reception timestamps associated to packets that were received by a single anchor. The resulting (filtered) dataset, sketched in Fig. 5.2(b), will represent the input for the proposed algorithm.

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

The following notation is introduced:

- T the duration of the measurement interval.
- $t_m \in [0, T]$ the transmission time (according to an absolute reference clock) of the m th packet transmitted by node $i(m)$.
- $\mathcal{A} \stackrel{\text{def}}{=} \{1, \dots, N\}$ the set of all anchor nodes.
- $\mathcal{A}_m \subseteq \mathcal{A}$ the set of anchor nodes that have successfully received and timestamped packet m . As the source node cannot hear its own packet, $i(m) \notin \mathcal{A}_m$.
- $N_m \stackrel{\text{def}}{=} |\mathcal{A}_m|$ the number of receiving anchors for packet m . Owing to the pre-filtering stage it holds that no less than two elements are included in this set, i.e., $N_m \geq 2$.
- $i(m) \in \{0, \dots, N\}$ denotes the transmitting node index for the generic m th packet.
- $\bar{n}_m \in \mathcal{A}_m$ the “reference anchor” for packet m , chosen randomly among the elements of $\mathcal{A}_m$.
- R_{nm} the reception timestamp of the m th packet at the receiving node n as measured by its local clock.
- e_{nm} a random error term (measurement noise) affecting the reception timestamp R_{nm} .
- θ_n the (unknown) clock offset term for node n .
- γ_n the (unknown) clock skew factor for node n , i.e., the relative difference between the actual and nominal clock frequency.
- $\mathbf{p}_n \stackrel{\text{def}}{=} [x_n, y_n, z_n]$ the position of generic anchor node $n = 1, \dots, N$ in the reference 3D coordinate system.
- $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v} \cdot t$ the instantaneous position of the blind node at time t .
- $\mathbf{v} \stackrel{\text{def}}{=} [v_x, v_y, v_z]$ the velocity vector of the blind node in the reference 3D coordinate system.

- $d_{k,n} \stackrel{\text{def}}{=} \|\mathbf{p}_k - \mathbf{p}_n\| = d_{n,k}$ the Euclidean distances between nodes k and n .

For the sake of more compact notation the variables t_m , $m = 1, \dots, M$ are stacked into the vector $\mathbf{t}$, and the variables θ_n and γ_n , $n = 1, \dots, N$ are stacked into the vectors $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$.

The work assumes that the clock of the generic node n is affected by two systematic error terms: a temporal offset θ_n and a (relative) frequency offset γ_n .

Furthermore, timestamps are affected by i.i.d. random measurement errors. A simplified scenario is considered, where signal propagation occurs through direct Line-of-Sight (LOS) path between any pair of nodes. The 3D position of all anchor nodes are known exactly without error. The blind node position is assumed as constrained on the horizontal plane (hence $v_z = 0$) at known height z_0 , and the problem is to determine the horizontal components $[x_0, y_0, v_x, v_y]$. With the above notation, the reception timestamp measured at anchor n for the m th packet sent by another anchor $i(m) \neq n$ writes as (anchor-to-anchor equation):

$$(1 + \gamma_n) R_{nm} = t_m + \frac{d_{n,i(m)}}{c} + \theta_n + e_{nm}, \quad (5.1)$$

where $d_{n,i(m)} \stackrel{\text{def}}{=} \|\mathbf{p}_{i(m)} - \mathbf{p}_n\|$ denotes the (known) distance between the transmitting and receiving anchors, and c the speed of light. If instead the m th packet is transmitted by the blind node, the reception timestamp at the generic anchor n writes as (blind-to-anchor equation):

$$(1 + \gamma_n) R_{nm} = t_m + \frac{1}{c} \|\mathbf{p}_0 + \mathbf{v} \cdot t_m - \mathbf{p}_n\| + \theta_n + e_{nm}. \quad (5.2)$$

The overall resolution method relies on the model equations (5.1) and (5.2) and is sketched in Fig. 5.3. The contribution devises a two-stage approach, where synchronization and localization (or tracing) are performed by two distinct routines. First, clock error terms $\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}}$ are estimated from anchor-anchor measurements, with the method presented later in Sec. 5.5, and used to correct the blind-anchor timestamps. The adjusted timestamp for a generic blind-anchor timestamp will be denoted by

$$R'_{nm} \stackrel{\text{def}}{=} (1 + \hat{\gamma}_n) R_{nm} - \hat{\theta}_n$$

In the second stage, the adjusted timestamps R'_{nm} are used to determine $\mathbf{p}_0, \mathbf{v}$, with the algorithms developed later in Sec. 5.6.

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

We conclude this discussion with a few remarks about the motivations that have driven the choice of a two-stages approach. In fact, it is possible in principle to join the estimation of clock error parameters and trajectory parameters into a single larger estimation instance (an approach followed e.g. by [156, 159] in other system scenarios). While this approach might seem more compact, it would come at the cost of higher resolution complexity. In other words, separating synchronization from localization lowers the overall computation cost. Also, in extended scenarios where multiple blind nodes are to be traced in parallel, separation allows to run the synchronization instance only once, and then reuse the clock error term estimates for all the localization instances associated to different blind nodes.

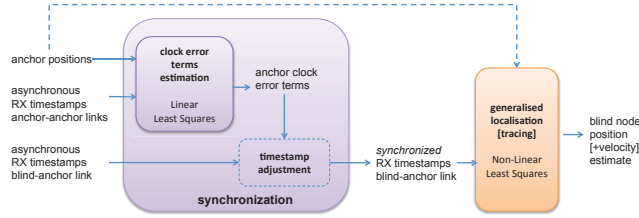


Figure 5.3: Workflow of the proposed method.

5.5 Estimation of Clock Error Terms - Synchronization

The first stage of the devised approach follows a procedure similar in spirit to the synchronization method proposed earlier in [153] for a different system model, with the necessary adaption to suit our scenario.

It is possible to note that the generic anchor-to-anchor equation (5.1) is linear into the unknown variables θ_n , γ_n ($n = 1, \dots, N$) and t_m ($m = 1, \dots, G$). Therefore the estimation problem can be casted into a linear LS problem, for which a closed form solution can be computed directly. Careful analysis reveals, however, that the problem is under-determined, i.e., it does not have a unique solution (see discussion in [153, Sec. VI]). To remove the ambiguity, it is sufficient to pick one anchor to serve as clock reference: without loss of generality we pick the first anchor with $n = 1$ as reference, and set $\hat{\theta}_1 = 0$ and $\hat{\gamma}_1 = 0$ as fixed parameters. Therefore we are left with $2(N - 1)$ clock error terms plus G unknown transmission times to estimate.

5.5 Estimation of Clock Error Terms - Synchronization

For a more compact matrix notation, the variable vector $\mathbf{y}$ of length $2(N-1) + G$ can be defined, as:

$$\mathbf{y} \stackrel{\text{def}}{=} [\theta_2, \dots, \theta_N, \gamma_2, \dots, \gamma_N, t_1, \dots, t_G]^T.$$

In order to represent the available measurements in matrix form, all possible GN measurements (for G packets at N anchors) are mapped to a single index $k = (m-1)N + n$, with $m = 1, \dots, G$ and $n = 1, \dots, N$. Denote by $\mathbf{b}$ the vector of length GN whose generic element is defined by ($k = (m-1)N + n$):

$$b_k \stackrel{\text{def}}{=} \begin{cases} d_{n,i(m)}/c - R_{nm} & \text{if } n \in \mathcal{A}_m \\ 0 & \text{otherwise} \end{cases}$$

wherein $k = (m-1)N + n$. Denote by $\mathbf{H}$ the matrix of size $GN \times 2(N-1) + G$ with elements h_{kj} defined as

$$h_{kj} \stackrel{\text{def}}{=} \begin{cases} -1 & \text{if } j = n, n \geq 2, n \in \mathcal{A}_m \\ R_{nm} & \text{if } j = N + n - 2, n \geq 2, n \in \mathcal{A}_m \\ -1 & \text{if } j = 2N - 2 + m \text{ and } n \in \mathcal{A}_m \\ 0 & \text{otherwise} \end{cases}$$

wherein again $k = (m-1)N + n$. With the above positions, and considering that all timestamp errors are assumed i.i.d., the linear LS estimate derived from the model equations (5.1) writes as:

$$\hat{\mathbf{y}} = \arg \min_{\mathbf{y}} (\mathbf{H}\mathbf{y} - \mathbf{b})^T (\mathbf{H}\mathbf{y} - \mathbf{b}) = \mathbf{H}^+ \mathbf{b}, \quad (5.3)$$

wherein $\mathbf{H}^+$ denotes the Moore-Penrose pseudo-inverse of $\mathbf{H}$.

After estimating the clock error terms $\hat{\theta}_n, \hat{\gamma}_n$ from anchor-to-anchor measurements, all reception timestamps for blind-to-anchor packets can be corrected. The generic adjusted timestamp will be denoted by

$$R'_{nm} \stackrel{\text{def}}{=} \hat{\gamma}_n \cdot R_{nm} - \hat{\theta}_n. \quad (5.4)$$

This procedure is equivalent to align all anchor clocks to the reference anchor clock, leaving only an (unknown) offset between the common clock frequency (equal for all anchors after timestamp adjustment) and the nominal clock frequency. It can be shown that the impact of such a residual common offset onto the final position estimate is negligible in practice. More in detail, [3, Appendix A] shows that the additional relative

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

error on the final position is of the same magnitude as the relative frequency tolerance of the receiving device, that is typically in the order of a few tens of ppm for commercial-off-the-shelf devices, hence absolutely negligible for any practical application.

5.6 Generalized Localization Algorithms

The second stage of the devoted approach uses the adjusted timestamps from the first phase to estimate the initial position and the velocity vector of the target moving blind node. In what follows, three possible equivalent approaches are described in detail.

5.6.1 Preliminaries

Replacing the raw timestamps with the adjusted timestamps (5.4) after synchronization, the blind-to-anchor equation (5.2) rewrites as:

$$R'_{nm} = t_m + \frac{1}{c} \|\mathbf{p}_0 + \mathbf{v} \cdot t_m - \mathbf{p}_n\| + e_{nm}. \quad (5.5)$$

Based on this model equation, it is possible to cast the tracing problem into a Non-linear LS form. However, before proceeding further it is also possible to introduce a small but important simplification in the basic model equation (5.5): it is the approximation of the (unknown) packet transmission time appearing in the distance term by the (known) arrival time at some reference anchor. For every packet m , one of the receiving anchors is arbitrarily picked to serve as reference. The terminology $\bar{n}_m \in \mathcal{A}_m$ denotes the reference anchor selected for packet m , and $\bar{R}'_m \stackrel{\text{def}}{=} R'_{nm}|_{n=\bar{n}_m}$ denotes the associated (adjusted) reception timestamp. After replacing t_m with $\bar{R}'_m$ in the distance term, eq. (5.5) rewrites as:

$$R'_{nm} \cong t_m + \frac{1}{c} \|\mathbf{p}_0 + \mathbf{v} \cdot \bar{R}'_m - \mathbf{p}_n\| + e_{nm}. \quad (5.6)$$

Thanks to the above simplification, the transmission time t_m now appears only as an additive term and therefore can be easily eliminated, as we show below, with positive effects on the resolution process. On the other hand, the negative impact on the quality of the final estimate is absolutely negligible. In fact, the error introduced by approximating $t_m \approx \bar{R}'_m$ into the distance term is in the order of $\frac{|v|}{c}$ times the distance between the blind node and the reference anchor, and can be safely neglected in practical applications. For example, in case of a blind node moving at speed of

$|v| = 10$ m/s (corresponding to 36 km/h) at 300 meters distance from the reference anchor, the approximation error is in the order of 10^{-5} meters.

For the sake of simpler notation, it is convenient to rewrite the basic equation (5.6) as:

$$f_{nm}(\mathbf{z}) + t_m + e_{nm} = 0, \quad (5.7)$$

wherein the vector $\mathbf{z}$ gathers all the variables of interest:

$$\mathbf{z} \stackrel{\text{def}}{=} [\mathbf{p}_0, \mathbf{v}] = [x_0, y_0, v_x, v_y]$$

and the parametric function $f_{nm}(\mathbf{z})$ is defined as

$$f_{nm}(\mathbf{z}) \stackrel{\text{def}}{=} \frac{1}{c} \|\mathbf{p}_0 + \mathbf{v} \cdot \overline{R'_m} - \mathbf{p}_n\| - R'_{nm}. \quad (5.8)$$

5.6.2 TOA formulation

In the first TOA variant the (unknown) transmission times $\mathbf{t}$ are treated as nuisance variables to be estimated jointly with blind node position and velocity. From (5.7) it is possible cast the estimation problem into the following Non-linear LS form:

$$\begin{aligned} \hat{\mathbf{z}}, \hat{\mathbf{t}} &= \arg \min_{\mathbf{z}, \mathbf{t}} \sum_{m=1}^M \sum_{n \in \mathcal{A}_m} e_{nm}^2 \\ &= \arg \min_{\mathbf{z}, \mathbf{t}} \sum_{m=1}^M \sum_{n \in \mathcal{A}_m} (f_{nm}(\mathbf{z}) + t_m)^2 \\ &= \arg \min_{\mathbf{p}_0, \mathbf{v}, \mathbf{t}} \sum_{m=1}^M \sum_{n \in \mathcal{A}_m} \left(\frac{\|\mathbf{p}_0 + \mathbf{v} \cdot \overline{R'_m} - \mathbf{p}_n\|}{c} - R'_{nm} + t_m \right)^2. \end{aligned} \quad (5.9)$$

wherein it is assumed that the error terms e_{nm} associated to distinct reception events are (i) independent and (ii) have constant variance¹.

5.6.3 TDOA formulation

In the alternative TDOA variant the (unknown) transmission time t_m is eliminated by taking the difference of arrival times at different anchors. However, subtraction introduces correlation between the residuals, hence the need to consider non-diagonal

¹This assumption is largely justified in practical applications based on COTS devices. In fact, the reception timestamp reported by commercial radio receivers is typically mapped (at best) to the first sample of the detected preamble. Consequently, timestamp resolution is limited by the ADC sampling rate of the radio front-end. This limit is coarser than the theoretically achievable timestamp accuracy and, differently from the latter, it is independent from the Signal-to-Noise Ratio of the arriving signal

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

covariance matrix. For a generic packet m sent by the blind node, consider the reference anchor $\bar{n}_m$ and another receiving anchor $n \neq \bar{n}_m$, and denote the difference between their respective TOA residuals by:

$$\begin{aligned}\xi_{nm} &\stackrel{\text{def}}{=} e_{nm} - e_{\bar{n}_m m} \\ &= f_{nm}(\mathbf{z}) - \bar{f}_m(\mathbf{z}) \\ &= \frac{\|\mathbf{p}_0 + \mathbf{v} \cdot \bar{\mathbf{R}}'_m - \mathbf{p}_n\| - \|\mathbf{p}_0 + \mathbf{v} \cdot \bar{\mathbf{R}}'_m - \mathbf{p}_{\bar{n}_m}\|}{c} - R'_{nm} + \bar{R}'_m.\end{aligned}\tag{5.10}$$

wherein $\bar{f}_m(\mathbf{z}) \stackrel{\text{def}}{=} f_{\bar{n}_m m}(\mathbf{z})|_{n=\bar{n}_m}$.

In other words, ξ_{nm} represents the TDOA residual for packet m associated to the pair receiving anchors n and $\bar{n}_m$. For the sake of notational simplicity, all TDOA residuals for packet m are stacked into the vector $\boldsymbol{\xi}_m \stackrel{\text{def}}{=} \{\xi_{nm}, n \in \mathcal{A}_m \setminus \bar{n}_m\}$.

Owing to the presence of a common term, the components of vector $\boldsymbol{\xi}_m$ are not independent, and therefore the (generalized) LS form must take into account a non-diagonal covariance matrix. Let $\boldsymbol{\Sigma}_m$ denote the covariance matrix of $\boldsymbol{\xi}_m$. It can be easily recognized that $\boldsymbol{\Sigma}_m$ has a particularly simple structure, independent of the choice of reference anchor, with all diagonal elements equal to '2' and all off-diagonal elements equal to '1':

$$\boldsymbol{\Sigma}_m = \begin{bmatrix} 2 & 1 & 1 & \dots & 1 \\ 1 & 2 & 1 & \dots & 1 \\ & & \dots & & \\ 1 & 1 & 1 & \dots & 2 \end{bmatrix} = \mathbf{I} + \mathbf{O}_{[N_m-1]},$$

wherein the symbol $\mathbf{O}_{[k]}$ denotes the rank-one matrix of size $k \times k$ with all elements equal to '1'. By the Sherman-Morrison theorem [162] the inverse develops as:

$$\boldsymbol{\Sigma}_m^{-1} \stackrel{\text{def}}{=} (\mathbf{I} + \mathbf{O}_{[N_m-1]})^{-1} = \mathbf{I} - \frac{1}{N_m} \mathbf{O}_{[N_m-1]}.$$

Thanks to such a simple structure of the inverse covariance, the TDOA LS estimator can be developed into the following simple form (**TDOA formulation**):

$$\begin{aligned}\hat{\mathbf{z}} &= \arg \min_{\mathbf{z}} \sum_{m=1}^M \boldsymbol{\xi}_m^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\xi}_m \\ &= \arg \min_{\mathbf{z}} \sum_{m=1}^M \left(\sum_{\substack{n \in \mathcal{A}_m \\ n \neq \bar{n}_m}} \xi_{nm}^2 - \frac{1}{N_m} \left(\sum_{\substack{n \in \mathcal{A}_m \\ n \neq \bar{n}_m}} \xi_{nm} \right)^2 \right).\end{aligned}\tag{5.11}$$

Note that the search space of (5.11) has lower dimensionality than (5.9), since the nuisance variables $\mathbf{t}$ have been eliminated. Regarding the variables of interest $\mathbf{z}$, we will show below that TOA and TDOA estimators represented respectively by (5.9) and (5.11) *lead exactly to the same solution* $\hat{\mathbf{z}}$ when instantiated with the same input data.

5.6.4 S-TOA formulation

Starting from the highly dimensional TOA formulation (5.9), it is possible to derive an alternative formulation, with the same dimensionality of the TDOA form (5.11), but without the need of introducing pairwise timestamp differences. This new formulation will be denoted with the term *shrunk TOA* (S-TOA for short).

Recall that the solution $\hat{\mathbf{z}}$ to the unconstrained minimization (5.9) must satisfy the zero-gradient condition, therefore the derivative of the objective function with respect to the generic transmission time t_m must be null, i.e., $\frac{\partial}{\partial \tau_i} \sum_{m=1}^M \sum_{n \in \mathcal{A}_m} (f_{nm}(\mathbf{z}) + t_m)^2 = 0$ for $i = 1, \dots, M$. This condition develops as:

$$\begin{aligned} \frac{\partial}{\partial \tau_i} \sum_{n \in \mathcal{A}_m} (f_{nm}(\mathbf{z}) + t_m)^2 &= 0 \\ \rightarrow \sum_{n \in \mathcal{A}_m} (f_{nm}(\mathbf{z}) + t_m) &= 0 \\ \rightarrow t_m &= -\frac{1}{N_m} \sum_{n \in \mathcal{A}_m} f_{nm}(\mathbf{z}), \quad m = 1, \dots, M. \end{aligned} \quad (5.12)$$

Since the solution of problem (5.9) must necessarily satisfy (5.12), including this constraint explicitly into the problem will not change the solution in the space of $\mathbf{z}$. This can be implemented by simply replacing in the objective function the variable t_m with the last term of (5.12). In this way, after some trivial passages (5.9) develops as:

$$\begin{aligned} \hat{\mathbf{z}} &= \arg \min_{\mathbf{z}} \sum_{m=1}^M \sum_{n \in \mathcal{A}_m} \left(f_{nm}(\mathbf{z}) - \frac{1}{N_m} \sum_{n \in \mathcal{A}_m} f_{nm}(\mathbf{z}) \right)^2 \\ &= \arg \min_{\mathbf{z}} \sum_{m=1}^M \left(\sum_{n \in \mathcal{A}_m} f_{nm}^2(\mathbf{z}) - \frac{1}{N_m} \left(\sum_{n \in \mathcal{A}_m} f_{nm}(\mathbf{z}) \right)^2 \right). \end{aligned} \quad (5.13)$$

Note that the variable t_m have been effectively eliminated, thus reducing the dimensionality of the search space from $K + M$ down to K (i.e., the length of the variable vector $\mathbf{z}$).

5.6.5 Equality of TDOA and S-TOA

Comparing (5.13) to (5.11), it is evident a strong similarity between the objective functions of S-TOA and TDOA. Recalling the definition of ξ_{mn} from (5.10), both estimators can be written in the following common format:

$$\hat{\mathbf{z}} = \arg \min_{\mathbf{z}} \sum_{m=1}^M v(g_{1m}, \dots, g_{N_m m})$$

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

with the operator $v()$ defined by

$$v(g_1, \dots, g_N) \stackrel{\text{def}}{=} \sum_{n=1}^N g_n^2 - \frac{1}{N} \left(\sum_{n=1}^N g_n \right)^2, \quad (5.14)$$

and¹

$$g_{nm} \stackrel{\text{def}}{=} \begin{cases} f_{nm}(\mathbf{z}) - \bar{f}_m(\mathbf{z}) & \text{for TDOA,} \\ f_{nm}(\mathbf{z}) & \text{for S-TOA.} \end{cases} \quad (5.15)$$

for $n = 1, \dots, N$. The term $v(g_1, \dots, g_N)$ in (5.14) represents the sum of squared deviations from the mean for the set $\{g_n, n = 1, \dots, N\}$, and rescaling by $\frac{1}{N-1}$ would lead to the canonical expression for the sample variance. This is not coincidental, since the LS method can be indeed interpreted as the minimization of the sample variance among the available measurements. Likewise the sample variance, $v()$ is invariant to translation: the addition of a common term μ to all sample components does not change the value of (5.14), formally:

$$v(g_1, g_2, \dots, g_N) = v(g_1 - \mu, g_2 - \mu, \dots, g_N - \mu), \quad \forall \mu$$

as can be easily verified with a few simple algebraic passages. From (5.15) it is possible to immediately recognize that this is indeed the case when comparing TDOA and S-TOA, with $\bar{f}_m(\mathbf{z})$ in (5.15) playing the role of the common shift μ . Therefore, TDOA and S-TOA yield the same value of the objective function for any generic point $\mathbf{z}$ of the search space.

5.6.6 On the equivalence between the TOA and TDOA/S-TOA

In summary, the discussion in previous subsections have shown that:

- The S-TOA (5.13) and TDOA (5.11) have equal objective functions and are defined over the same low-dimensional search space. In other words, they are two ways of expressing the same estimator, which will be denoted as TDOA/S-TOA hereafter.

¹Note that $\xi_{nm}|_{n=\bar{n}_m} = 0$, hence the inner sum term in (5.11) can be extended to the whole set $\mathcal{A}_m$, i.e. $\sum_{n \in \mathcal{A}_m, n \neq \bar{n}_m} \xi_{nm} = \sum_{n \in \mathcal{A}_m} \xi_{nm}$.

- By construction TOA and S-TOA are equivalent in the sense that they lead to the same solution in the (sub)space of the variable of interest $\mathbf{z}$. However, the TOA formulation (5.9) has larger dimensionality, with $M + K$ total variables instead of only K , due to the transmission time of every packet acting as nuisance variable. In other words, we may regard the TOA form as a sort of “expansion” of the TDOA/S-TOA estimator over a larger dimensionality space.

Recall that both forms are instances of continuous non-linear optimization problems, to be solved numerically. Therefore, while both forms would theoretically lead to the same solution, the “cost” of computing this solution is not necessarily the same. At this point it is natural to ask which form bears the least resolution cost

5.6.7 A first look at resolution cost

For this work, all non-linear minimization instances have been implemented and solved with a general purpose solver, namely the `fminunc` function that is implemented natively in Matlab[©]. This procedure uses a trust-region method to minimize the input function¹. In all resolution instances, the gradient was provided explicitly, but not the Hessian. All tolerance parameters were set to the same value for all estimators. As expected based on the discussion above, all estimators always converged to the same solution. However, they did so through different search paths.

In order to provide a first glimpse about the relative computation weight of the two forms, Fig. 5.4 reports the distribution of the *number of iterations* needed to solve each estimator form in one sample simulation scenario (details about the scenario are given later in Sec. 5.7.1) for 1000 MonteCarlo trials.

As expected, the curves for S-TOA and TDOA overlap exactly: in fact, the two objective functions (and their gradients) are equal across the whole search space, and therefore the numerical procedure follows exactly the same search path for the two estimators.

A bit surprisingly, we found that TOA requires considerably less iterations to converge to the final solution than TDOA/S-TOA. In 95% of the trials TOA converged to the final solution in less than a dozen iterations, whereas TDOA/S-TOA required up to 80 iterations. Other tests with different parameter settings and scenarios (not

¹See <http://it.mathworks.com/help/optim/ug/fminunc.html>

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

reported here) consistently confirmed this finding: it took systematically less iterations to TOA to converge to the final solution than TDOA/S-TOA.

This result came unexpected, considering that TOA goes over a search space of much higher dimensionality than TDOA/S-TOA, with $M + K$ variables versus only K (in this specific scenario $M = 40$ and $K = 4$). In some sense, it seems that the *expansion* of the search space with additional additive terms is helping the iterative process to converge in less steps. On the other hand, this gain is partially offset by the heavier computation involved by TOA at each iteration, both in terms of time and memory space. For this reason, it is not possible to express a preference of general validity between TOA and TDOA/S-TOA, and the choice between the two should be based on the particular application context and computation platform¹.

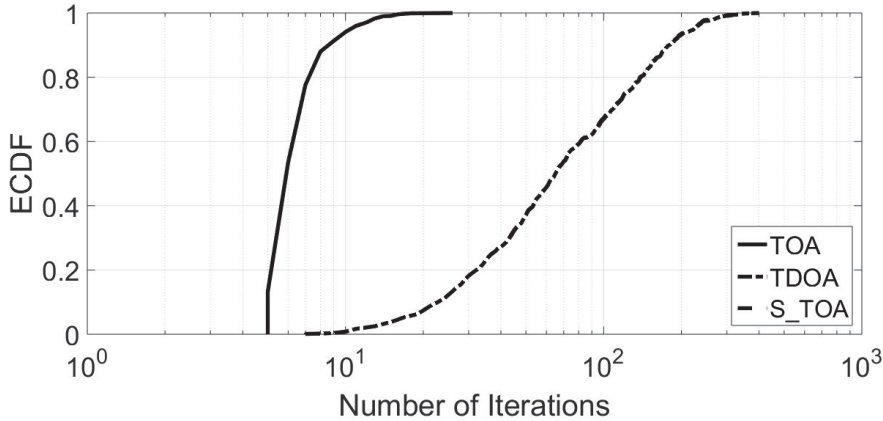


Figure 5.4: Number of iterations of dynamic models of TOA, TDOA, and S-TOA algorithms in the asynchronous case with 20% packet losses.

¹The definition of *cost* and the most relevant factors thereof (time, memory, communication overhead) depend on the particular computation environment and application context, e.g., whether the computation is carried out in hardware or software, by a centralized entity or in a distributed fashion, etc. Given a particular reference scenario, in principle it is possible to design and fine-tune numerical resolution strategies specifically tailored to these particular instances. In other words, when it comes to comparing performance of numerical methods, it would be more appropriate to compare the performance of different *implementations* of the proposed estimators with reference to particular scenarios. The in-depth investigation of these aspects remains outside the scope of the present contribution and is left for future follow-up work.

5.6.8 Particularization to a static emitter (localization)

The previous sections developed a small set of estimators for the generalized localization problem (also called *tracing* in [145]), where the unknown velocity $\mathbf{v}$ of a (linearly) moving node must be estimated in addition to the initial position $\mathbf{p}_0$. As all estimators ultimately lead to exactly the same solution, it is possible to refer to them collectively as “TOA/TDOA” estimators hereafter. In the particular case that the blind node is known to be static, the tracing problem reduces to the classic *localization* problem. It is possible to have two different options to address the (static) localization problem with our TOA/TDOA estimators:

- **Dynamic TOA/TDOA.** With this approach we use any of the estimators derived above, taking velocity $\mathbf{v}$ as an unknown variable. The solution will in general deliver a small but non-zero¹ velocity estimate $\hat{\mathbf{v}}$. The mid-point of the estimated trajectory $\hat{\mathbf{p}}_0 + \frac{1}{2}\hat{\mathbf{v}}T$ is then selected as the estimate of the (fixed) node position.
- **Static TOA/TDOA.** With this approach the information about the node being static is encoded *ex ante* into the estimator. Any of the the estimators derived above can be used, but their formulation is particularized by setting $\mathbf{v} = 0$ into the function $f_{mn}()$ defined in (5.8), leaving the position $\hat{\mathbf{p}}_0$ as the only variable of interest to be determined. In other words, (5.8) is replaced by the following:

$$f_{nm}(\mathbf{z}) \stackrel{\text{def}}{=} \frac{1}{c} \|\mathbf{p}_0 - \mathbf{p}_n\| - R'_{nm}. \quad (5.16)$$

Alternatively, it is possible to resort to some of the previously published method. For the sake of comparison, for the particular case of fixed blind node the following sections compares our approach with an alternative method previously proposed in [3], hereafter referred as “range-based DTDOA”. A schematic workflow of this method is given in Fig. 5.5. A key point of this method is represented by the the estimation of pseudo-ranges (or equivalently: distance differences) between the blind node and the anchors, hence the term “range-based”. Timestamps are used to estimate ranges, and the latter are then used to estimate positions. This strategy has two drawbacks. First, it does not extend easily to the case of moving node, where the range (distance) between the blind node and each anchor varies. Second, it breaks the estimation process from

¹In some sense, the estimator will “interpret” part of the measurement noise as node movement.

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

timestamp measurements to final position estimate, by introducing an intermediate layer of variables to be determined (i.e., the pseudo-ranges). This represents a sort of approximation of the original problem, and therefore comes with a certain loss of optimality.

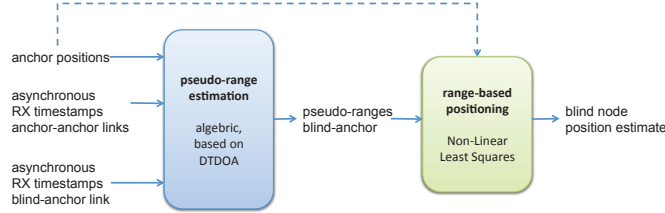


Figure 5.5: Workflow of the range-based DTDOA method in [3].

5.7 Simulation analysis

In this section the estimators formulated in Section 5.6 will be applied to a simple simulation scenario.

5.7.1 Scenario

The blind node lies within a squared area of interest of size 250×250 meters, at the (known) fixed height of $z_0 = 1.5$ meters. For each trial, the initial blind node position $\mathbf{p}_0$ is extracted randomly within the area of interest and its speed is set randomly between 3 and 6 m/s. The moving direction is random, with the constraint of ensuring that the final point remains within the area of interest. We consider $N = 5$ anchors placed as depicted in Fig. 5.6 at the fixed height of 4 meters. The relative frequency offset and temporal offset of each node clock are extracted randomly in the range $[-40, +40]$ ppm and $[0, 100]$ ms, respectively. Timestamp measurement errors are i.i.d. uniformly distributed in $[-\tau, +\tau]$, with $\tau = 25$ ns (corresponding to 20 MHz sampling rate). The reception of a given packet at a generic anchor occurs with probability $1 - \ell$ independently from other reception events of other packets and anchors, therefore ℓ represents the packet loss probability. During the observation period of duration $T = 10$ sec, each anchor node emits $B = 10$ beacons while the blind node emits $M = 40$ packets.

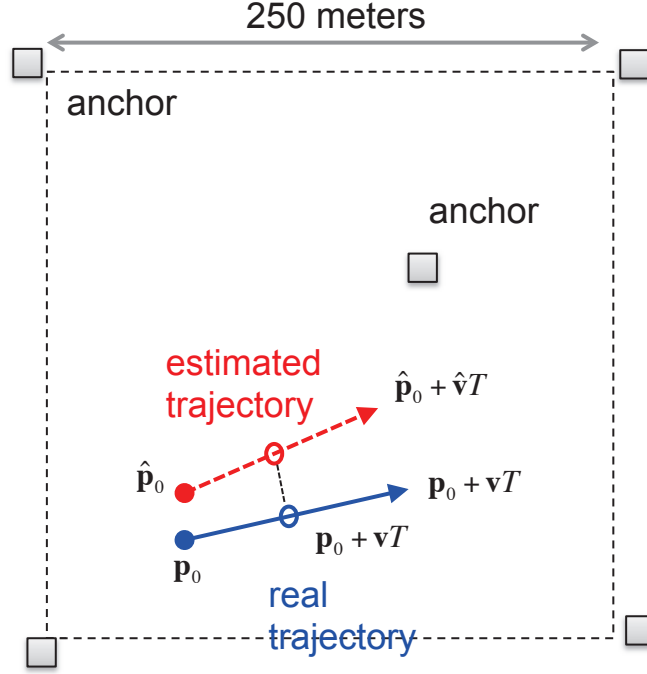


Figure 5.6: Reference topology with $N = 5$ anchors.

5.7.2 Comparison with range-based DTDOA for static emitter

This subsection compares the accuracy of the proposed method against the range-based DTDOA method from [3], in the particular case of a static blind node. Fig. 5.7 reports the error distribution for the three methods described earlier in Sec. 5.6.8 for the case of $\ell = 0$. In fact, the lossless scenario is the most favorable case for the DTDOA method, wherein input measurements are built from the difference of four reception timestamps associated to *packet pairs*, and is therefore more sensitive to packet loss. Despite the favorable condition, the range-based DTDOA performs markedly worse than the direct TOA/TDOA methods proposed in this work. Note also that the accuracy of the Dynamic TOA/TDOA approach is very close to that of the Static TOA/TDOA.

5.7.3 Results with moving node

In order to compare the estimated and actual trajectories the following error metrics have been considered:

- The distance ϵ_{mid} between the mid-points of the estimated and actual trajectories,

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

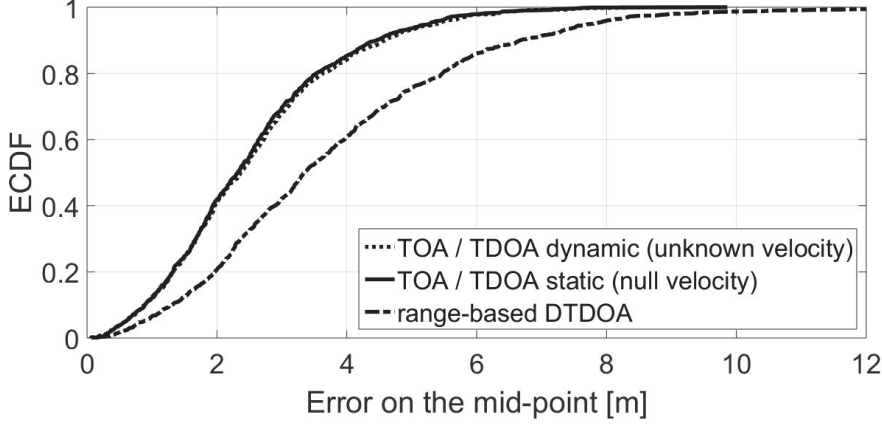


Figure 5.7: Final localization error for static blind node (asynchronous scenario with $\ell = 0$).

i.e.

$$\epsilon_{mid} = \|\hat{\mathbf{p}}_0 - \mathbf{p}_0 + \frac{1}{2}(\hat{\mathbf{v}} - \mathbf{v})T\|$$

- The difference ϵ_{speed} between the estimated and actual speed:

$$\epsilon_{speed} = \|\hat{\mathbf{v}}\| - \|\mathbf{v}\|$$

- The angle ϵ_{angle} between the estimated and actual velocity vector

$$\epsilon_{angle} = \arccos \frac{\hat{\mathbf{v}}^T \mathbf{v}}{\|\hat{\mathbf{v}}\| \|\mathbf{v}\|}$$

Fig. 5.8(a) reports the distribution of the mid-point error ϵ_{mid} obtained with the three different estimators and for different values of packet loss intensities $\ell \in \{0, 0.2, 0.4\}$. As predicted by the analysis in the previous section, in each trial the three estimators TOA, TDOA and S-TOA lead to the same solution (up to small shifts in the order of 0.1 mm due to numerical approximations) and therefore the corresponding error curves overlap. As expected, increasing the packet loss implies a certain loss of information, and therefore lower accuracy. While for moderate packet loss such a degradation remains contained (compare the curves for $\ell = 0$ and $\ell = 0.2$), it becomes noticeable for the higher loss value.

The curves in Fig. 5.8(a) were obtained in the case of asynchronous anchors (with independent clocks) after estimating the clock error terms with the procedure described

in Sec. 5.5 and adjusting the reception timestamps accordingly. It is interesting to compare these results with Fig. 5.8(b), reporting what is obtained in the ideal scenario where all receiving anchors are perfectly synchronized *ex ante* to a common clock, and therefore no timestamp adjustment is needed. Comparing the corresponding curves in Fig. 5.8(a) and Fig. 5.8(b) it is possible to observe that the penalty due imperfect recovery of clock synchronization is pretty small (less than half meter in this scenario). These results indicate that synchronization, if not provided natively by the system, can be effectively achieved in post-processing based on timing measurements from reference transmitters (not necessarily cooperative). On the other hand, the penalty due to high packet loss is much higher, since high packet loss translates directly into loss of data.

From a system engineering point of view, the provided results indicate that it would be more convenient for the sake of localization accuracy to invest in better reception, with smarter receivers that are able to reduce packet loss (lower ℓ) and at the same time increase timing accuracy (lower measurement noise variance), rather than investing in building additional synchronization infrastructure.

Figure 5.9 reports the estimated speed (i.e., the modulus of velocity vector) versus the true speed of the blind node. It can be seen that in the tested scenario the speed can be estimated within ± 0.5 m/s, and that the speed error is rather independent from the actual speed.

Fig. 5.10 reports the (absolute) error on the estimated velocity angle. In the tested scenario, the estimated velocity direction was mostly within $\pm 5^\circ$ off the actual direction, and only in rare cases exceeded $\pm 10^\circ$.

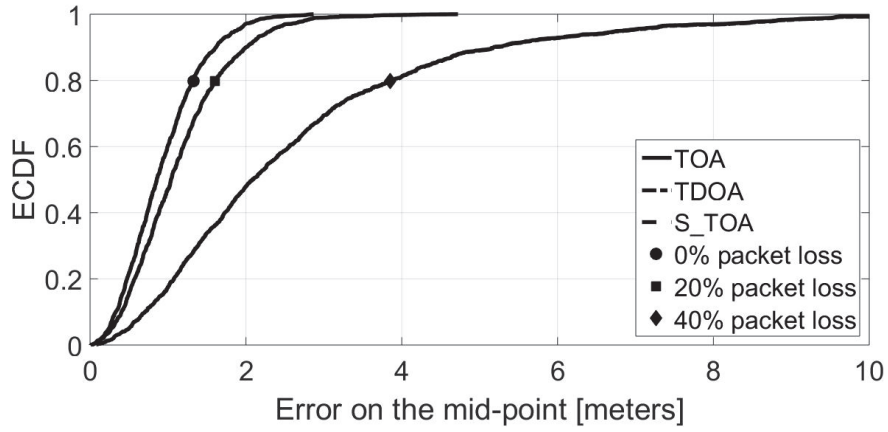
5.8 Experimental Results

In order to validate the proposed solution in a real environment, some experiments have been performed in a testbed deployed at University of Brescia, and composed of commercial-off-the-shelf WiFi devices.

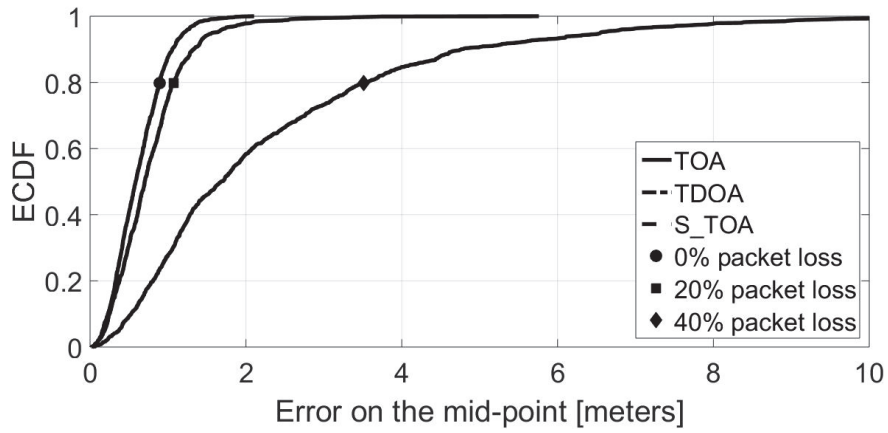
5.8.1 Testbed setup

The testbed deployment consists of multiple fixed devices (anchors), plus one mobile device acting as blind node.

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES



(a) Asynchronous anchors



(b) Synchronous anchors

Figure 5.8: ECDF of horizontal error on mid-point for different loss intensities ($\ell \in \{0, 0.2, 0.4\}$).

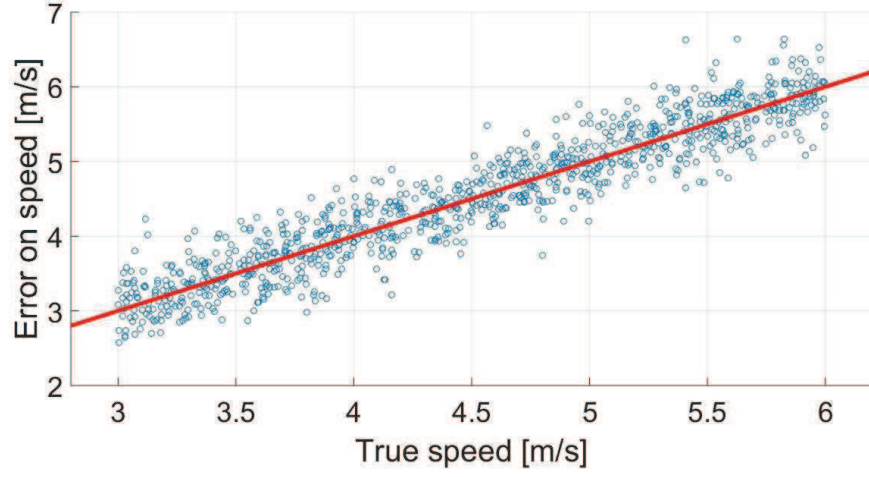


Figure 5.9: Module of the estimated speed versus module of the true speed in the asynchronous case with 20% packet losses.

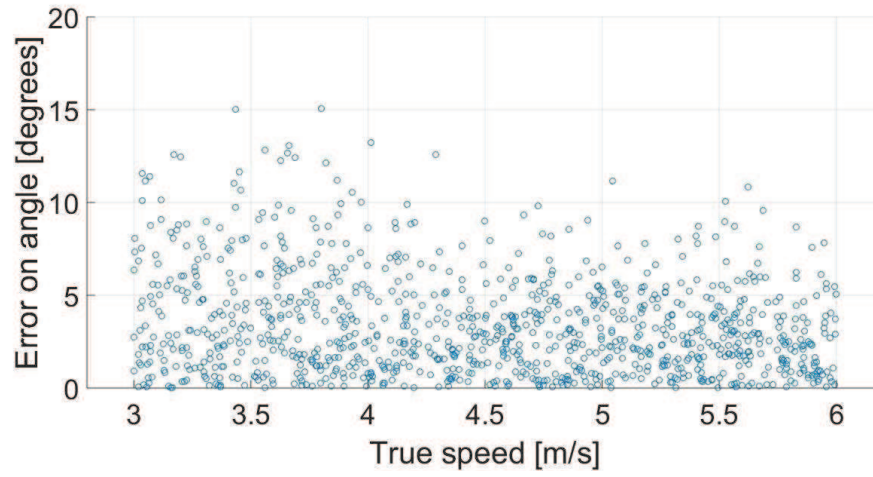


Figure 5.10: Angle between the vector of estimated speed and the vector of true speed versus module of the true speed in the asynchronous case with 20% packet losses.

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

Transmitting nodes are programmed to send UDP packets with a payload of 10 bytes at a configurable average rate. Every packet is transmitted with a single attempt at the MAC layer, i.e., retransmissions are disabled in order to avoid ambiguities in matching copies of the same packet received by different nodes. In a real scenario this is equivalent to consider only the first transmission of a frame (retry bit set to 0). Channel access at the MAC layer is performed using the standard Distributed Coordination Function (DCF) protocol with a Contention Window of 15 slots. All nodes are set to transmit in IEEE 802.11g mode with Orthogonal Frequency Division Multiplexing (OFDM) modulation, transmission rate of 6Mb/s.

The measurement testbed includes WRT54GL devices from Linksys, the same used in [3]. These devices are cheap (less than 40 USD on ebay) and run a very robust OpenWRT distribution based on Linux Kernel 2.6.32. Most importantly, their Network Interface Card (NIC) is compatible with OpenFWWF [163], an open source firmware that replaces the original binary-only software from Broadcom and has been widely used as research platform [164, 165, 166]. The basic version of OpenFWWF implements a fully working DCF and timestamps each incoming packet with the value of the Basic Service Set clock, sampled by the underlying hardware when a preamble is successfully received. Unfortunately this clock has only 1 MHz resolution. For this reason the real-time receiving part of the code has been modified to spin during the reception of specific packets until they are completely decoded: right after the single instruction spin-lock, a sample of the 32 bit internal clock that drives the execution pipeline is taken and reported to the Linux kernel. In this way it is possible to extract the hardware timestamp (according to the local clock) corresponding to the first sample of the incoming packet. Although the hardware clock runs at 88 MHz, the last two bits of every timestamps are non-informative, since the timing resolution is limited by the ADC sampling rate of $F_s = 20$ MHz. As a result, raw timestamps come with a resolution of $T_s = 1/F_s = 50$ ns, corresponding to 15 meters at speed light. For the first two sets of experiments we used the raw timestamps obtained in this way.

All experiments were conducted in an open outdoor area (garden) with all nodes in full Line-of-Sight (LOS) conditions. The actual node positions (ground truth) were measured manually with an accuracy in the order of 2-3 centimetres. All transmissions took place on a single frequency channel (specifically, channel #1).

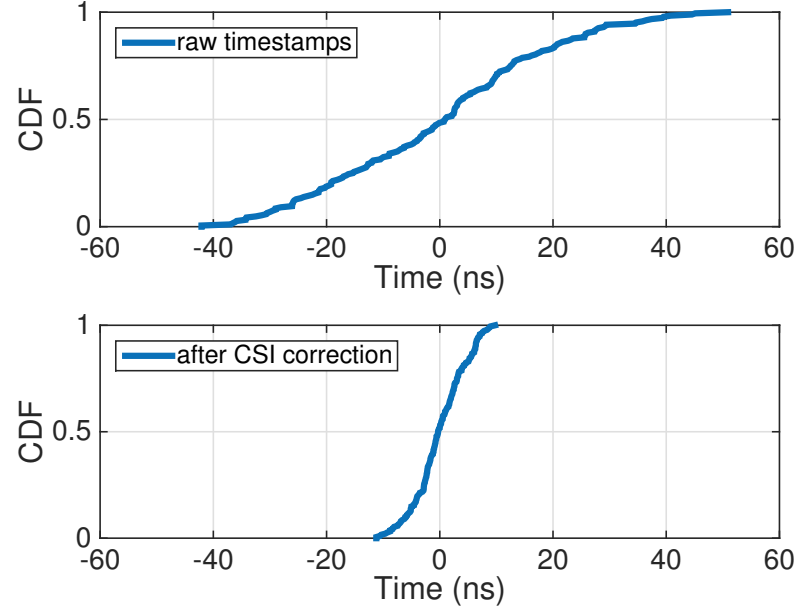


Figure 5.11: ECDF of time-of-arrival differences between two receiving anchors after compensating for clock drift and offset. Top plot: raw timestamps before CSI correction. Bottom plot: after CSI correction.

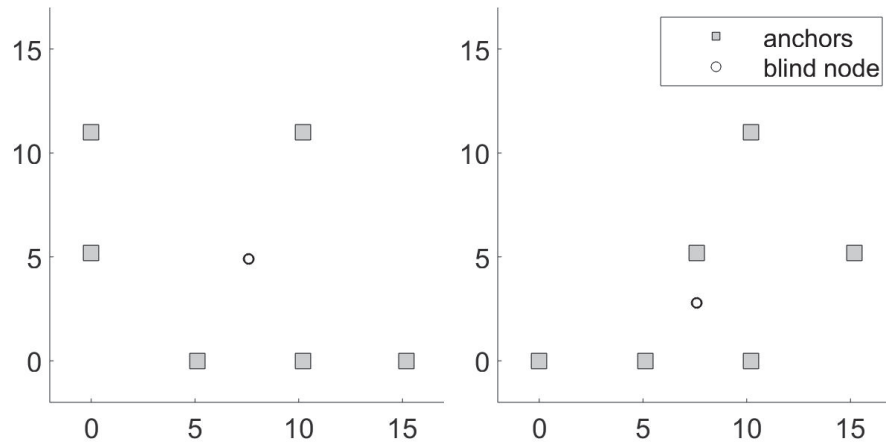


Figure 5.12: Two of the 6-anchor topologies used for comparison with range-based DT-DOA in WiFi testbed experiments.

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

5.8.2 Experiment #1: comparison with range-based DTDOA localization for a static node

In the first set of experiments the aim is to compare the accuracy of the proposed estimators with the range-based DTDOA method described earlier in [3], for the particular scenario of static blind node. These tests considered three different topologies consisting of 6 anchors and one (static) blind node placed in a rectangular area of 15×10 meters. Two of the tested topologies are shown in Fig. 5.12. For each topology 18 different datasets have been collected, by repeating the measurement in different (non overlapping) time intervals. In this way it is possible to consider a total of 54 different trials. The duration of each trial was 3 second, and each node (anchor and blind) was set to transmit at the average rate of 55 packets/second.

The ECDF of the absolute error plotted in Fig. 5.13 confirms that the proposed direct resolution methods (range-free) beats the range-based DTDOA method, a result that is in full agreement with the previous simulation results from Fig. 5.7.

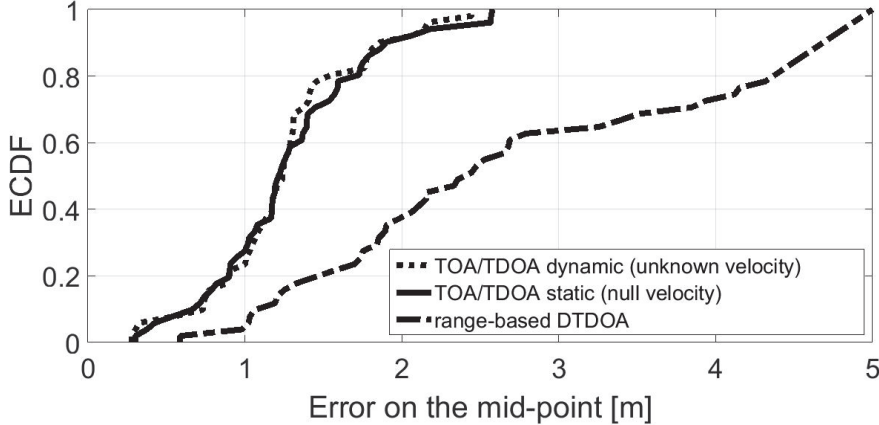


Figure 5.13: Experimental performances of different localization schemes for static node: ECDF of absolute error over 54 trials.

5.8.3 Experiment #2: moving node

The second set of experiments considered a fixed topology of 10 anchors. For each trial, a person was walking at (approximately) constant speed along a straight trajectory fully contained within the convex hull of the anchor nodes. Each individual experiment

lasts between 4 and 5 seconds. During each trial, each anchor and the blind node was transmitting packets at the average rate of 56 packets/sec.

Fig. 5.14 reports the actual trajectory (ground truth, in blue continuous line) along with the final estimated trajectory (dashed red line) for some sample experiments. It can be seen that the proposed method is able to estimate closely the trajectory, with mid-point error always below 1 meter.

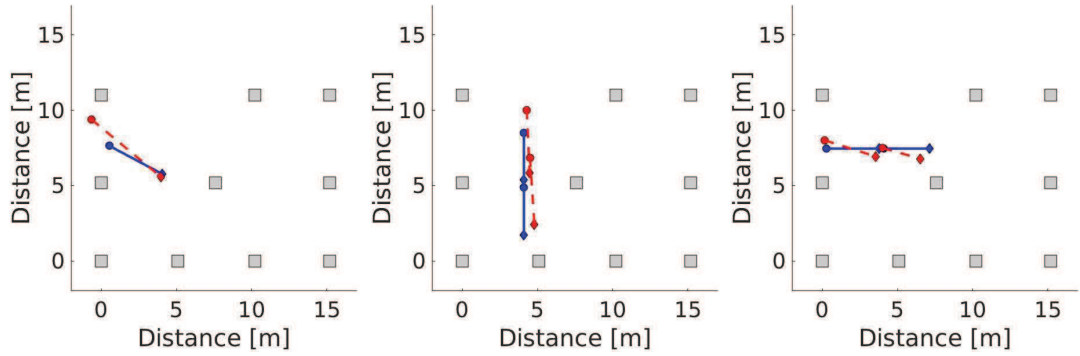


Figure 5.14: Actual trajectory (blue) vs. Estimated trajectory (red) with TOA in three sample experiments.

5.8.4 Experiment #3: moving node with refined timestamps

The third set of experiments adopted a new prototype version of the OpenFWWF firmware that is able to report for each received packet also the Channel State Information (CSI), i.e., an estimate of the channel frequency response computed by the baseband processor on the basis of the 52-bits preamble. The baseband processor uses the CSI to equalize the OFDM symbol, but this information can be used to improve the resolution of the time-of-arrival measurement and achieve sub-sample accuracy, as elaborated in the recent work [167]. Inspired by the latter, a method for timestamp correction based on CSI has been implemented, that is similar in spirit to the method proposed in [167], but particularized for a single channel. In this way it is possible to improve the timing resolution by a factor $\times 5$. To illustrate, Fig. 5.11 plots the ECDF of the difference between the time-of-arrival of the same packet measured by two distinct receiving anchors, after estimating and compensating for the relative clock offset and skew. The top and bottom plot refer, respectively, to the estimated time-of-arrival difference before and after CSI correction. The distribution stretches in the range ± 50

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

ns (i.e., $\pm T_s$) for the raw timestamps (top plot), while after CSI correction (bottom plot) it reduces down to approximately ± 10 ns, corresponding to 3 meters at speed light.

For this third round of experiments 9 fixed nodes have been placed in a regular 3×3 grid as shown in Fig. 5.15, plus one mobile device acting as blind node. Differently from the previous experiments, where each anchor was serving both as receiver and (reference) transmitter, in this round each anchor has a specialized role: the 8 external anchors were configured to serve purely as receivers, while the single central node (denoted as “pivot” node hereafter) serves as the single reference transmitter for the synchronization phase.

Both the pivot and the blind node act purely as transmitters and do not cooperate to the localization process. They are programmed to send UDP packets with a payload of 10 bytes at the average rate of 9 packet/second. The time-of-arrival measurements (after CSI correction) associated to the packets transmitted during this period by the pivot and blind node are given in input, respectively, to the synchronization phase (Eq. (5.3)) and to the TOA estimator (5.9).

Fig. 5.15 reports the actual and estimated trajectory for three sample experiments. The accuracy is roughly similar to the previous set of experiments, with mid-point errors below 1 meter, but it should be noted that this result was achieved with a much smaller number of packets compared to the previous experiments. In other words, the improved timing resolution gained by means of CSI correction allows to use a smaller number of more accurate measurements. Moreover, with the new timestamps also the estimated error on the initial and the final point is below 1 meter of accuracy.

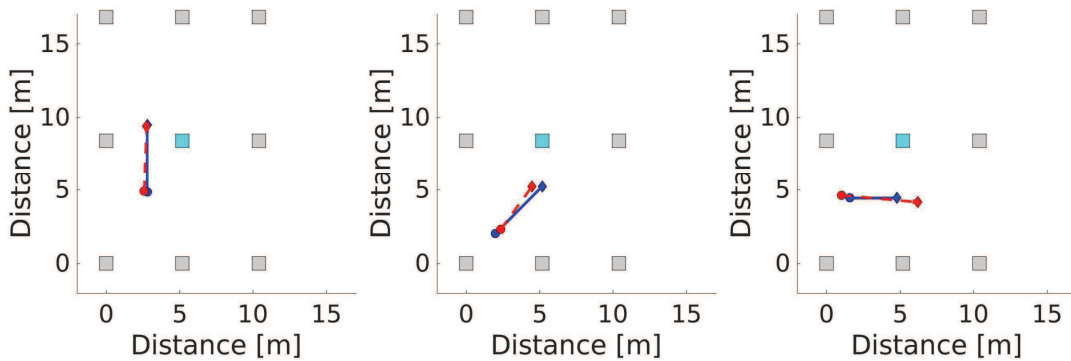


Figure 5.15: Actual trajectory (blue) vs. Estimated trajectory (red) with TOA in three sample experiments with CSI correction.

5.9 Migration to IoT devices

The solution developed in the previous sections have been tested in Wi-Fi real devices. However, the solution is particularly suitable also for IoT devices.

When dealing with localization of IoT nodes, in fact, advanced localization mechanisms such as GPS become unfeasible, because of the hard computation and energy requirements posed by these systems [168].

The asynchronous localization scheme proposed in the previous sections requires only a radio able to transmit/receive packets and the capability to download the timestamps gathered in the localization phase to a server, able to start off-line computations. For these reasons, it seems a providing solution to be applied also in the IoT technology.

From the other side, the limitations of the devices involved in the IoT has to be considered carefully. In particular, the sampling rate of the radio ADC and the clock frequency of the particular device are critical parameters to be considered.

The performance evaluation of the proposed method using real IoT devices is left as a future work. However, the aforementioned features of the devoted localization technique, as well as simulation results obtained in Section 5.7, stem as a promising solution.

5. TIME-BASED WIRELESS LOCALIZATION TECHNIQUES

6

Conclusions and Future Research Directions

This PhD work has considered the problem of providing advances services for IoT applications, ranging from security mechanisms to localization functionalities.

First of all, the work demonstrates that times are nearly mature to endow IoT devices and deployments with advanced security functionalities, both at the MAC layer and the application layer.

With reference to wide set of security threats described in Chapter 2, in Chapter 3 this PhD work provides experimental indications about the possibility to leverage hardware functionalities to speed up the execution of a well-known security algorithm, as the AES protocol, to embed confidentiality and integrity services at the MAC layer of the IEEE 802.15.4-2015 technology. Moreover, Chapter 3 describes also an innovative framework, offering: (i) a rich set of possible security configurations that can be enabled also in heterogeneous scenarios where protected communications may coexist with unsecured ones; (ii) an effective methodology for easing the initialization of secure IEEE 802.15.4 networks; (iii) a lightweight KMP for negotiating a layer-2 key between a node pair; (iv) resilience to several security attacks, and (v) a full compatibility with IEEE 802.15.4 technology.

In this context, Physical-layer security techniques have been also studied and a concrete proposal, leveraging the wireless channel intrinsic anonymity, have been implemented in real devices and experimentally evaluated.

6. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

Moving the target on the application layer, Chapter 4 provides technical solutions for the management of end-to-end security. Specifically, a lightweight KMP is described which provides strong authentication and key agreement in IoT devices minimal air-time consumption, thanks to the use of implicit ECQV certificates. Moreover, the basic authentication and authorization functionalities tailored for a federation of IoT platforms and studied with the EU project symbIoTe has been described. Such innovative architecture is currently under implementation and has reached the consensus of all involved partners of the symbIoTe consortium.

From the works previously discussed, it emerges that the nowadays' technology is really able to provide security services for IoT networks. Specifically, the possibility to leverage dedicated hardware registers and routines can significantly speed up the execution of such processes, without affecting the normal operations of the IoT network. The choice of a particular security protocol or mechanism depends not only on the security level that is required, but also on other related performances indexes, such as throughput or latency, that are directly impacted by the presence of security countermeasures. When devices with harder constraints are considered, physical-layer security techniques represent a valid option. However, the resilience of such solutions with respect to the active attacks and the relationship of this techniques with the emerging radio fingerprinting solutions must be carefully investigated.

As regarding future directions of this research activities, deeper tests will be performed in order to understand the impact of security functionalities at different layers of the protocol stack, as well as potential issues derived by their joint adoption in real use-cases.

Finally, with the aim of providing lightweight localization and tracking techniques suitable for the integration in constrained IoT devices, Chapter 5 formalizes the tracing/localization problem within a specific system model (valid both for Wi-Fi and IoT deployments) and presents a possible resolution strategy. Specifically, it provides different variants of LS estimators based on TOA and TDOA, by proving formally their equivalence and generalizing previous results about TOA and TDOA equivalence. Numerical results from Monte Carlo simulations are provided to explore the impact of various system-level aspects and parameters. Finally, results are validated on experimental data from a real-world WiFi testbed, paving the way to the experimental assessment on a real IoT deployment.

The implementation and integration of security and localization functionalities in real IoT applications and scenarios is the next practical step that will be tackled in next month and goes beyond what is included this PhD Work. Moreover, starting from the algorithms developed in this work, specialized customization will be studied, developed and experimentally evaluated for more general use-cases, as planes localization and indoor tracing. The integration process could further help to understand the limits of the proposed mechanisms in live deployments and demonstrating adaptation capabilities with respect to other real-time services tailored for IoT applications.

6. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

References

- [1] M. Campagna, “SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV),” Certicom Research, Technical Report, Jan. 2013. xii, 146
- [2] K. Podnar Zarko, I. Katsaros, M. Kusek, E. Garrido, M. Pardi, J. Garcia, A. Antonic, Z. Zelenika, A. Jaworski, M. Pilc, C. Ruggenthaler, and L. de Santis, “Deliverable d1.2,” 2016, project deliverable symbIoTe. xii, 166, 167
- [3] N. Facchi *et al.*, “Emitter localisation from reception timestamps in asynchronous networks,” *Computer Networks*, vol. 88, September 2015. xiii, 184, 186, 191, 199, 200, 201, 206, 208
- [4] O. Hersent, D. Boswarthick, and O. Elloumi, *The Internet of Things: Key Applications and Protocols*. Wiley, 2012. 1
- [5] S. Sciancalepore, M. Vucinic, G. Piro, G. Boggia, and T. Watteyne, “Link-layer Security in TSCH networks: effect on slot duration,” *Transactions on Emerging Telecommunications Technologies (ETT)*, vol. 27, Aug. 2016. [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1002/ett.3089/full> 1, 24, 25, 71, 72, 75, 76
- [6] L. A. Grieco, A. Rizzo, S. Colucci, S. the, G. Piro, D. Di Paola, and G. Boggia, “IoT-aided robotics applications: technological implications, target domains and open issues,” vol. 54, December 2014. 1
- [7] D. Evans, “The internet of things, how the next evolution of the internet is changing everything,” Cisco Internet Business Solutions Group (IBSG). White paper, Apr. 2011. 1
- [8] Ericsson, “More than 50 billion connected devices,” Ericsson White Paper, Feb. 2011. 1

REFERENCES

- [9] B. Emmerson, “M2M: the Internet of 50 billion devices,” *Huawei Win-Win Magazine Journal*, no. 4, pp. 19–22, Jan. 2010. 1
- [10] M. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. Grieco, G. Boggia, and M. Dohler, “Standardized Protocol Stack for the Internet of (Important) Things,” *IEEE Commun. Surveys Tuts*, 2012. 2, 3
- [11] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, “Internet of Things: Vision, Applications & Research Challenges,” *Ad Hoc Networks*, 2012. 2
- [12] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010. 2
- [13] K. Ashton, “That Internet of Things Thing,” *RFID Journal*, 2009. 2
- [14] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. S. Kwak, “The Internet of Things for Health Care: A Comprehensive Survey,” *IEEE Access*, vol. 3, pp. 678–708, 2015. 4
- [15] D. Niewolny, “How the Internet of Things Is Revolutionizing Healthcare,” NXP Freescale White Paper, Oct. 2013. 5
- [16] “Energy management based on internet of things: practices and framework for adoption in production management,” *Journal of Cleaner Production*, vol. 100, pp. 235 – 246, 2015. 5
- [17] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, “On optimal scheduling in duty-cycled industrial iot applications using ieee802.15.4e tsch,” *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3655–3666, Oct 2013. 6, 8, 60
- [18] H. Foundation, *HART Communication Protocol and Foundation*, Standard, HART Std., 2012. 7
- [19] S. Petersen and S. Carlsen, “WirelessHART Versus ISA100.11a: The Format War Hits the Factory Floor,” *IEEE Industrial Electronics Magazine*, vol. 5, no. 4, pp. 23–34, Dec 2011. 7, 46

REFERENCES

- [20] I. Foundation, *Wireless Systems for Industrial Automation: Process Control and Related Applications*, Standard ISA-100.11a, ISA Std., 2009. 7
- [21] *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*, IEEE Std., 16 Apr. 2012. 8, 15, 17, 78, 81, 92
- [22] “IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs),” *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, Apr. 2016. 8, 29, 55, 58, 65, 71, 92, 124
- [23] *Zigbee Specification*, Zigbee Alliance Std., 17 Jan. 2008. 10
- [24] *802.15.4-2011: IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Std., 16 Jun. 2011. 10, 71, 92
- [25] S. Sciancalepore, G. Piro, G. Boggia, and L. A. Grieco, “Application of IEEE 802.15.4 security procedures in OpenWSN protocol stack,” *IEEE Standards Education e-Magazine*, vol. 4, 4th Quarter 2014. 12, 18, 28, 29, 34, 66, 76, 105
- [26] N. Accettura and G. Piro, “Optimal and Secure Protocols in the IETF 6TiSCH communication stack,” in *Proc. of IEEE International Symposium on Industrial Electronics (ISIE)*, Jun. 2014. 13, 20, 21
- [27] L. Harte, *Introduction to Bluetooth*. Althos, 2009. 13, 48
- [28] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, “Standardized Protocol Stack for the Internet of (Important) Things,” *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1389–1406, Third 2013. 15, 16, 17, 22
- [29] P. Thubert and T. Watteyne, “IPv6 over the TSCH mode of IEEE 802.15.4e (6tisch),” IETF, WG charter. [Online]. Available: <https://datatracker.ietf.org/wg/6tisch/charter/> 19
- [30] Q. Wang and X. Villajosana, “6top Protocol (6P) draft-ietf-6tisch-6top-protocol-03,” IETF, Internet Draft, Oct. 2016. 19

REFERENCES

- [31] P. Thubert, “An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4 draft-ietf-6tisch-architecture-10,” IETF, Internet Draft, Jun. 2016. 20
- [32] M. Richardson, “6tisch Secure Join protocol draft-richardson-6tisch-dtsecurity-secure-join-00,” IETF, Internet Draft, Sep. 2016. [Online]. Available: <https://datatracker.ietf.org/doc/draft-richardson-6tisch-dtsecurity-secure-join/> 20, 61
- [33] X. Villajosana and K. Pister, “Minimal 6TiSCH Configuration, draft-ietf-6tisch-minimal-17,” IETF, Internet Draft, Nov. 2017. 21, 106, 131, 165
- [34] N. Kushalnagar, G. Montenegro, and C. Schumacher, *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*, RFC 4919, Internet Engineering Task Force RFC 4919, Aug. 2007. 21
- [35] G. Montenegro, N. Kushalnagar, , G. Hui, and D. Culler, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, RFC 4944, Internet Engineering Task Force RFC 4944, Sep. 2007. 21
- [36] S. Park, K. Kim, , S. Chakrabarti, and J. Laganier, *IPv6 over Low Power WPAN Security Analysis draft-daniel-6lowpan-security-analysis-05*, IETF DRAFT, Internet Engineering Task Force DRAFT, Mar. 2011. 21
- [37] P. Thubert, A. Brandt, , J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, RFC 6550, Internet Engineering Task Force RFC 6550, Mar. 2012. 21
- [38] J. Postel, *User Datagram Protocol*, RFC 768, Internet Engineering Task Force RFC 768, Aug. 1980. 22
- [39] Z. Shelby, K. Hartke, , and C. Bormann, *The Constrained Application Protocol (CoAP)*, RFC 7252, Internet Engineering Task Force RFC 7252, Jun. 2014. 23
- [40] TelosB datasheet. Last Retrieved: 12/10/2016. [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf 24

REFERENCES

- [41] Zolertia Z1 datasheet. Last Retrieved: 12/10/2016. [Online]. Available: http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf 25
- [42] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, *OpenMote: Open-Source Prototyping Platform for the Industrial IoT*. Springer International Publishing, 2015, pp. 211–222. 25, 163
- [43] P. Boccadoro, M. Barile, G. Piro, and L. Grieco, “Energy consumption analysis of TSCH-enabled platforms for the Industrial-IoT,” in *2nd International Forum on Research and Technologies for Society and Industry (RTSI 2016)*, Sep. 2016. 26, 32
- [44] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. D. Glaser, and K. S. J. Pister, “OpenWSN: a Standards-Based Low-Power Wireless Development Environment,” *Trans. on Emerg. Telecom. Technol.*, vol. 23, no. 5, pp. 480–493, 2012. 26, 66, 87, 113, 163
- [45] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. C. Schmidt, “RIOT OS: Towards an OS for the Internet of Things,” in *IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2013, pp. 79–80. 32
- [46] W. Stallings, *Cryptography and Network Security - Principles and Practice*. Prentice Hall, 2005. 33, 38, 41, 42, 43, 44, 45
- [47] O. Garcia-Morchon, H. Kumar, S. Keoh, R. Hummen, and R. Struik, “Security Considerations in the IP-based Internet of Things draft-garcia-core-security-06,” IETF, Internet Draft, Sep. 2013. 35
- [48] J. Granjal, E. Monteiro, and J. S. Silva, “Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues,” *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, thirdquarter 2015. 38
- [49] W. Diffie and M. Hellman, “New Directions in Cryptography,” *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, Sep. 2006. 40, 43, 89, 121

REFERENCES

- [50] S. Sciancalepore, G. Piro, E. Vogli, G. Boggia, L. Grieco, and G. Cavone, “LICI-TUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT,” *Computer Networks (COMNET)*, Elsevier, vol. 108, pp. 66–77, Oct. 24 2016. 41, 51, 86, 87, 90, 113, 131
- [51] E. Barker, D. Johnson, and M. Smid, *Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, NIST Spec. Pub. 800-56a, National Institute for Standards and Technology (NIST) - Computer Security Division Std., Mar. 2007. 45
- [52] D. Huckelmann, M. Harrison, and F. Michahelles, *Architecting the Internet of Things*. Springer, 2011. 49
- [53] C. Wang, T. Jiang, and Q. Zhang, *ZigBee Network Protocols and Applications*. Auerbach Publications, 2014. 50, 87, 104
- [54] T. Kennedy and R. Hunt, “A Review of WPAN Security: Attacks and Prevention,” in *Proc. of Int. Conf. on Mobile Technol., Applications, and Syst.* ACM, 2008. 50
- [55] M. Vucinic, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, “OSCAR: Object Security Architecture for the Internet of Things,” in *Int. Symp. on a World of Wirel., Mobile and Multim. Netw. (WoWMoM)*. IEEE, Jun. 2014, pp. 1–10. 58
- [56] S. L. Keoh, S. Kumar, and H. Tschofenig, “Securing the Internet of Things: A Standardization Perspective,” *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, Jun. 2014. 58, 62, 144, 154
- [57] *AES-CCM Cipher Suites for Transport Layer Security (TLS)*, Internet Engineering Task Force Std. RFC6655, 2012. 58
- [58] *AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS*, Internet Engineering Task Force Std. RFC7251, 2014. 58
- [59] K. Rantos, A. Papanikolaou, and C. Maniavas, “IPsec over IEEE 802.15.4 for Low Power and Lossy Networks,” in *Int. Symp. on Mobility Manag. and Wirel. Acc. (MobiWac)*. New York, NY, USA: ACM, 2013, pp. 59–64. 58

-
- [60] *Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)*, Internet Engineering Task Force Std. RFC4309, 2005. 59
- [61] S. Raza, S. Duquennoy, J. Höglund, U. Roedig, and T. Voigt, “Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN,” *Security and Commun. Netw.*, Wiley, vol. 7, no. 12, pp. 2654–2668, Dec. 2014. 59
- [62] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig, “Securing Communication in 6LoWPAN with Compressed IPsec,” in *Distributed Computing in Sensor Systems and Workshops (DCOSS)*, June 2011, pp. 1–8. 59
- [63] D. Altolini, V. Lakkundi, N. Bui, C. Tapparello, and M. Rossi, “Low Power Link Layer Security for IoT: Implementation and Performance Analysis,” in *Int. Wirel. Commun. and Mobile Comput. Conf. (IWCMC)*, Jul. 2013, pp. 919–925. 59, 60, 71, 94
- [64] Y. W. Law, J. Doumen, and P. Hartel, “Survey and Benchmark of Block Ciphers for Wireless Sensor Networks,” *ACM Trans. Sen. Netw.*, vol. 2, no. 1, pp. 65–93, Feb. 2006. 59
- [65] M. Healy, T. Newe, and E. Lewis, *Smart Sensors and Sensing Technology*. Springer Berlin Heidelberg, 2008, ch. Analysis of Hardware Encryption Versus Software Encryption on Wireless Sensor Network Motes, pp. 3–14. 59
- [66] P. Hamalainen, M. Hannikainen, and T. Hamalainen, “Efficient Hardware Implementation of Security Processing for IEEE 802.15.4 Wireless Networks,” in *Midwest Symposium on Circuits and Systems*, vol. 1, Aug. 2005, pp. 484–487. 59
- [67] L. Huai, X. Zou, Z. Liu, and Y. Han, “An Energy-Efficient AES-CCM Implementation for IEEE802.15.4 Wireless Sensor Networks,” in *Int. Conf. on Netw. Security, Wirel. Commun. and Trusted Comput. (NSWCTC)*, vol. 2, Apr. 2009, pp. 394–397. 59
- [68] C. T. O. Otero, J. Tse, and R. Manohar, “AES Hardware-Software Co-design in WSN,” in *Int. Symp. on Asynchr. Circuits and Syst. (ASYNC)*. IEEE, May 2015, pp. 85–92. 59

REFERENCES

- [69] R. Daidone, G. Dini, and M. Tiloca, “On Experimentally Evaluating the Impact of Security on IEEE 802.15.4 networks,” in *Int. Conf. on Distr. Comput. in Sensor Systems and Workshops (DCOSS)*, Jun. 2011, pp. 1–6. 59, 71
- [70] F. Chen, X. Yin, R. German, and F. Dressler, “Performance Impact of and Protocol Interdependencies of IEEE 802.15.4 Security Mechanisms,” in *Int. Conf. on Mobile Adhoc and Sensor Syst. (MASS)*. IEEE, Oct. 2009. 59, 71
- [71] M. Vucinic, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, “Energy cost of security in an energy-harvested IEEE 802.15.4 Wireless Sensor Network,” in *Med. Conf. on Embedded Comp. (MECO)*, Jun. 2014, pp. 198–201. 59, 60, 71
- [72] Y. Xiao, H.-H. Chen, B. Sun, R. Wang, and S. Sethi, “MAC Security and Security Overhead Analysis in the IEEE 802.15.4 Wireless Sensor Networks,” *Eurasip Journal on Wirel. Comm. and Netw.*, pp. 1–12, May 2006. 59, 60, 71
- [73] H. Yan, Y. Zhang, Z. Pang, and L. D. Xu, “Superframe Planning and Access Latency of Slotted MAC for Industrial WSN in IoT Environment,” *IEEE Trans. on Ind. Inf.*, vol. 10, no. 2, pp. 1242–1251, May 2014. 60
- [74] T. Watteyne, L. Doherty, J. Simon, and K. Pister, “Technical Overview of SmartMesh IP,” in *Int. Worksh. on Ext. Seamlessly to the Internet of Things (esIoT)*, Taiwan, 3-5 July 2013. 60
- [75] D. Stanislawski, X. Vilajosana, Q. Wang, T. Watteyne, and K. Pister, “Adaptive Synchronization in IEEE802.15.4e Networks,” *IEEE Trans. on Ind. Inf.*, vol. 10, no. 1, pp. 795–802, Feb. 2014. 60
- [76] T. Watteyne, M. R. Palattella, and L. A. Grieco, *Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals*, RFC 7554, Internet Engineering Task Force RFC 7554, May 2015. 60, 88
- [77] R. Struik, “6TiSCH Security Architectural Elements, Desired Protocol Properties, and Framework draft-struik-6tisch-security-architecture-elements-01,” IETF, Internet Draft, Oct. 2014. 60

-
- [78] M. Richardson, “6tisch secure join using 6top draft-richardson-6tisch-security-6top-05,” IETF, Internet Draft, Nov. 2015. 60
- [79] M. Boujelben, O. Cheikhrouhou, M. Abid, and H. Youssef, “Establishing Pairwise Keys in Heterogeneous Two-Tiered Wireless Sensor Networks,” in *Proc. of Int. Conf. on Sensor Technologies and Applications, (SENSORCOMM)*, Jun. 2009, pp. 442–448. 61
- [80] M. Pawlowski, A. Jara, and M. Ogorzalek, “Extending Extensible Authentication Protocol over IEEE 802.15.4 Networks,” in *Int. Conf. on Innovative Mob. and Internet Serv. in Ubiquitous Comp. (IMIS)*, Jul. 2014, pp. 340–345.
- [81] Y. Li, “Design of a Key Establishment Protocol for Smart Home Energy Management System,” in *Int. Conf. on Comput. Intelligence, Commun. Sys. and Netw.(CICSyN)*, Jun. 2013, pp. 88–93. 61
- [82] M. Rahman, S. Sampalli, and S. Hussain, “A robust pair-wise and group key management protocol for wireless sensor network,” in *Proc. of IEEE GLOBECOM Workshops (GC Wkshps)*, Dec. 2010, pp. 1528–1532. 61
- [83] R. D. Pietro and G. Oligeri, “COKE Crypto-Less Over-the-Air Key Establishment,” *IEEE Trans. on Inf. Forensics and Security*, vol. 8, no. 1, pp. 163–173, Jan. 2013. 61
- [84] B. Tian, S. Han, S. Parvin, and T. S. Dillon, “A Key Management Protocol for Multiphase Hierarchical Wireless Sensor Networks,” in *Proc. of IEEE/IFIP Int. Conf. on Emb. and Ubiq. Comp. (EUC)*, Dec. 2010, pp. 617–623. 61
- [85] M. Wilhelm, I. Martinovic, and J. Schmitt, “Secure key generation in sensor networks based on frequency-selective channels,” *IEEE Journal on Sel. Areas in Comm. (JSAC)*, vol. 31, no. 9, pp. 1779–1790, Sep. 2013. 61
- [86] S. Raza, D. Tratalza, and T. Voigt, “Lightweight IKEv2: A Key Management Solution for both the Compressed IPsec and the IEEE 802.15.4 Security,” *Workshop on Smart Object Security, Paris*, Mar. 2012.

REFERENCES

- [87] W. Gu, N. Dutta, S. Chellappan, and X. Bai, “Providing End-to-End Secure Communications in Wireless Sensor Networks,” *IEEE Trans. on Netw. and Service Manag.*, vol. 8, no. 3, pp. 205–218, Sep. 2011.
- [88] J. Misić, “Cost of secure sensing in IEEE 802.15.4 networks,” *IEEE Trans. on Wireless Comm.*, vol. 8, no. 5, pp. 2494–2504, May 2009. 61
- [89] S. Sciancalepore, G. Piro, E. Vogli, G. Boggia, and L. A. Grieco, “On securing IEEE 802.15.4 networks through a standard compliant framework,” in *EuroMed Telco Conference 2014*, Naples, IT, Nov. 2014. 61, 88, 105
- [90] S. Sciancalepore, A. Capossole, G. Piro, G. Boggia, and G. Bianchi, “Key Management Protocol with Implicit Certificates for IoT systems,” in *Proc. of ACM Int. Workshop on IoT challenges in Mobile and Industrial Syst. (IoT-Sys)*, Florence, IT, May 2015. 61, 131, 144, 145
- [91] “IEEE Recommended Practice for Transport of Key Management Protocol (KMP) Datagrams,” *IEEE Std 802.15.9-2016*, Aug 2016. 61, 144, 154
- [92] S. Sciancalepore, G. Piro, G. Boggia, and G. Bianchi, “Public Key Authentication and Key agreement in IoT devices with minimal airtime consumption,” *IEEE Embedded System Letters (ESL)*, vol. -, no. -, 2016, to be published. 62
- [93] A. Ukil, S. Bandyopadhyay, A. Bhattacharyya, A. Pal, and T. Bose, “Auth-lite: Lightweight m2m authentication reinforcing dtls for coap,” in *IEEE Int. Conf. on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, Mar. 2014, pp. 215–219. 62
- [94] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, “Lithe: Lightweight Secure CoAP for the Internet of Things,” *IEEE Sensors J.*, vol. 13, no. 10, pp. 3711–3720, 2013. 62
- [95] D. Hardt, “The OAuth 2.0 Authorization Framework,” IETF, RFC 6749, Oct. 2012. 62, 154, 155
- [96] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, and H. Tschofenig, “Authorization for the Internet of Things using OAuth 2.0 draft-ietf-ace-oauth-04,” IETF, Internet Draft, Oct. 2016. 62

-
- [97] M. Jones, H. Tschofenig, and S. Erdtman, “CBOR Web Token (CWT) draft-ietf-ace-cbor-web-token-01,” IETF, Internet Draft, Jul. 2016. 62
- [98] S. Emerson, Y. K. Choi, D. Y. Hwang, K. S. Kim, and K. H. Kim, “An OAuth based authentication mechanism for IoT networks,” in *Int. Conf. on Inform. and Commun. Techn. Convergence (ICTC)*, Oct. 2015, pp. 1072–1074. 62
- [99] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, “IoT-OAS: An OAuth-Based Authorization Service Architecture for Secure Services in IoT Scenarios,” *IEEE Sensors J.*, vol. 15, no. 2, pp. 1224–1234, Feb. 2015. 62
- [100] CC2420 datasheet. Last Retrieved: 12/10/2016. [Online]. Available: <https://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf> 74
- [101] Piro, G. and Boggia, G. and Grieco, L.A., *A standard compliant security framework for Low-power and Lossy Networks draft-piro-6tisch-security-issues-03 (work in progress)*, IETF 6TiSCH WG, Dec. 2014. 88
- [102] S. Galbraith, *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012. 89
- [103] R. Housley, W. Polk, W. Ford, and D. Solo, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 3280, Internet Engineering Task Force RFC 3280, April 2002. 93, 104
- [104] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, “Wireless sensor network security: A survey, in book chapter of security,” in *in Distr., Grid, and Pervasive Comput.*, Yang Xiao (Eds. CRC Press, 2007, pp. 0–849. 93, 100
- [105] N. Floissac and Y. L’Hyver, “From AES-128 to AES-192 and AES-256, How to Adapt Differential Fault Analysis Attacks on Key Expansion,” in *Proc. of Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, Sept. 2011, pp. 43–53. 99
- [106] J. Yan, A. Blackwell, R. Anderson, and A. Grant, “Password memorability and security: empirical results,” *Security Privacy, IEEE*, vol. 2, no. 5, pp. 25–31, Sep. 2004. 100

REFERENCES

- [107] Dierks, T. and Rescorla, E., *The Transport Layer Security Protocol Version 1.1*, IETF, Apr. 2006. 100, 172
- [108] A. K. Ranjan, V. Kumar, and M. Hussain, “Security analysis of TLS authentication,” in *Proc. of Int. Conf. on Contemporary Computing and Informatics (IC3I)*, Nov. 2014, pp. 1356–1360. 101, 149
- [109] A. Ferreira, R. Giustolisi, J.-L. Huynen, V. Koenig, and G. Lenzini, “Studies in Socio-technical Security Analysis: Authentication of Identities with TLS Certificates,” in *Proc. of IEEE Int. Conf. on Trust, Security and Privacy in Comput. and Commun. (TrustCom)*, Jul. 2013, pp. 1553–1558. 101
- [110] B. Blanchet, “Automatic Verification of Correspondences for Security Protocols,” *Journal of Computer Security*, vol. 17, no. 4, pp. 363–434, Jul. 2009. 101
- [111] M. Abadi, B. Blanchet, and C. Fournet, “Just Fast Keying in the Pi Calculus,” *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 3, Jul. 2007. 101
- [112] R. Küsters and T. Truderung, “Using ProVerif to Analyze Protocols with Diffie-Hellman Exponentiation,” in *IEEE Computer Security Foundations Symposium*, July 2009, pp. 157–171. 101
- [113] S. Asadi and H. S. Shahhoseini, “Formal security analysis of authentication in SNMPv3 protocol by an automated tool,” in *Int. Symp. on Telecomm. (IST)*, Nov 2012, pp. 1060–1064. 101
- [114] D. Chang, X. Chu, and G. Wei, “Analysis of the Security-Enhanced vTPM Migration Protocol Based on ProVerif,” in *Int. Conf. on Comp. and Information Sciences (ICCIS)*, Jun. 2013, pp. 1437–1440. 101
- [115] B. Blanchet, “Using Horn Clauses for Analyzing Security Protocols,” in *Formal Models and Techniques for Analyzing Security Protocols*, ser. Cryptology and Information Security Series, V. Cortier and S. Kremer, Eds. IOS Press, Mar. 2011, vol. 5, pp. 86–111. 101
- [116] S. Sciancalepore, G. Piro, F. Bruni, E. Nasca, G. Boggia, and L. Grieco, “An IoT-based measurement system for aerial vehicles,” in *Proc. of IEEE Int. Workshop on Metrology for Aerospace (MAS)*, Benevento, IT, Jun. 2015. 112

-
- [117] P. Barsocchi, G. Oligeri, and C. Soriente, “Shake: Single hash key establishment for resource constrained devices,” *Ad Hoc Networks*, 2012. 119
- [118] R. Di Pietro and G. Oligeri, “COKE: Crypto-less over-the-air key establishment,” *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 1, pp. 163–173, jan. 2013. 119, 120, 136
- [119] —, “Esc,” *Comput. Netw.*, vol. 84, no. C, pp. 46–63, Jun. 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.comnet.2015.04.006> 119, 120
- [120] S. Mirzadeh, H. Cruickshank, and R. Tafazolli, “Secure device pairing: A survey,” *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 17–40, 2014. 119, 144
- [121] C. Bormann, M. Ersue, and A. Keranen, *Terminology for Constrained-Node Networks*, RFC 7228, Internet Engineering Task Force RFC 7228, May 2014. 120, 144
- [122] T. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. 136
- [123] Texas Instruments, “CC2538 datasheet,” <http://www.openmote.com/shop/openmote-cc2538.html>, 2015. 140, 141
- [124] AllAboutBatteries.com, “AA-type battery storage,” <http://www.allaboutbatteries.com/Energy-tables.html>, 2011. 141
- [125] K. Liu, S. Primak, and X. Wang, “On secret key generation from multiple observations of wireless channels,” in *Communication Systems (ICCS), 2014 IEEE International Conference on*, Nov 2014, pp. 147–151. 141
- [126] J. Zhang, R. Woods, A. Marshall, and T. Q. Duong, “An effective key generation system using improved channel reciprocity,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 1727–1731. 141
- [127] W. Xi, X.-Y. Li, C. Qian, J. Han, S. Tang, J. Zhao, and K. Zhao, “Keep: Fast secret key extraction protocol for d2d communication,” in *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*, May 2014, pp. 350–359. 141

REFERENCES

- [128] M. Mossinger, B. Petschkuhn, J. Bauer, R. C. Staudemeyer, M. Wojcik, and H. C. Pohls, “Towards quantifying the cost of a secure IoT: Overhead and energy consumption of ECC signatures on an ARM-based device,” in *IEEE Int. WoWMoM Symp.*, Jun. 2016. 144
- [129] O. P. Pinol, S. Raza, J. Eriksson, and T. Voigt, “BSD-based elliptic curve cryptography for the open Internet of Things,” in *Int. Conf. on New Technol., Mobil. and Secur. (NTMS)*, Jul. 2015, pp. 1–5. 144
- [130] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 5280, Internet Engineering Task Force RFC 5280, May 2008. 144
- [131] “Explaining Implicit Certificates,” Certicom, Tech. Rep., 2004. 145
- [132] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*. Cambridge University Press, 1999. 145, 148, 149
- [133] D. R. Brown, R. Gallant, and S. A. Vanstone, “Provably secure implicit certificate schemes,” in *Financial Cryptography*. Springer, 2001, pp. 156–165. 146, 148
- [134] D. R. Brown, M. J. Campagna, and S. A. Vanstone, “Security of ECQV-Certified ECDSA Against Passive Adversaries,” *IACR Cryptology ePrint Archive*, 2009. 148
- [135] C. Bormann and Z. Shelby, “Block-wise transfers in CoAP - draft-ietf-core-block-18,” IETF, Internet Draft, Sep. 2015. 150
- [136] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, *Recommendation for Key Management - Part 1: General*, NIST Spec. Pub. 800-57, National Institute for Standards and Technology (NIST) - Computer Security Division Std., 2012. 150
- [137] ECRYPT, *Yearly Report on Algorithms and Keysizes*, 2015. 138, 150
- [138] SECG, *SEC 2: Recommended Elliptic Curve Domain Parameters version 2.0*, 2004. 150

-
- [139] *IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques*, IEEE P1363a, Institute of Electrical and Electronics Engineers Std., 2004. 150
 - [140] *Constrained RESTful Environments (CoRE) Link Format*, Internet Engineering Task Force Std. RFC6690, 2012. 154
 - [141] P. Hunt, J. Richer, W. Mills, P. Mishra, and H. Tschofenig, “OAuth 2.0 Proof-of-Possession (PoP) Security Architecture draft-ietf-oauth-pop-architecture-08.txt,” IETF, Internet Draft, Jul. 2016. 154, 156, 160
 - [142] M. Jones and D. Hardt, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” IETF, RFC 4950, May 2012. 156
 - [143] M. Jones, J. Bradley, and N. Sakimura, “JSON Web Token (JWT),” IETF, RFC 5719, May. 2015. 156, 177
 - [144] A. Birgisson, J. Gibbs Politz, U. Erlingsson, and M. Lentczner, “Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud,” in *Proc. of the Conf. on Network and Distributed System Security Symposium*, 2014. 176
 - [145] F. Ricciato, S. Sciancalepore, and G. Boggia, “Tracing a linearly moving node from asynchronous time-of-arrival measurements,” *IEEE Communications Letters*, Jun. 2016, to be published. 183, 184, 199
 - [146] Y. Wang, X. Ma, and G. Leus, “Robust time-based localization in asynchronous networks,” *IEEE Trans. on Signal Processing*, vol. 59, no. 9, 2011. 186
 - [147] R. M. Vaghefi and R. M. Buehrer, “Asynchronous time-of-arrival-based source localization,” in *IEEE ICASSP*, 2013. 186
 - [148] R. Kim, T. Ha, and D. Jung, “Tdoa localization for wireless networks with imperfect clock synchronization,” in *IEEE ICOIN*, 2014. 186
 - [149] B. Liu, H. Chen, Z. Zhong, and V. Poor, “Asymmetrical round trip based synchronization-free localization in large-scale underwater sensor networks,” *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, 2010. 186

REFERENCES

- [150] D. Zachariah, A. De Angelis, S. Dwivedi, and P. Haendel, “Schedule-based sequential localisation in asynchronous wireless networks,” *EURASIP Journal of Advances in Signal Processing*, 2014. 186
- [151] M. R. Gholami, S. Gezici, and E. G. Strom, “Range-based sensor node localization in the presence of unknown clock skews,” in *IEEE ICASSP*, 2013. 186
- [152] T. Li, A. Ekpenyong, and Y.-F. Huang, “Source localization and tracking using distributed asynchronous sensors,” *IEEE Transactions on Signal Processing*, vol. 54, no. 10, 2006. 186
- [153] R. Rajan and A. J. Van der Veen, “Joint ranging and synchronization for an anchorless network of mobile nodes,” *IEEE Trans. on Signal Processing*, vol. 63, no. 8, April 2015. 186, 190
- [154] K. Yang, G. Wang, and Z. Q. Luo, “Efficient convex relaxation methods for robust target localization by a sensor network using time differences of arrivals,” *IEEE Trans. on Signal Processing*, vol. 57, no. 7, July 2009. 186
- [155] R. M. Vaghefi and R. M. Buehrer, “Cooperative joint synchronization and localization in wireless sensor networks,” *IEEE Trans. on Signal Processing*, vol. 63, no. 14, July 2015. 186
- [156] O. Jean and A. Weiss, “Passive localization and synchronization using arbitrary signals,” *IEEE Trans. on Signal Processing*, vol. 62, no. 8, April 2014. 186, 190
- [157] A. Coluccia, F. Ricciato, and G. Ricci, “Positioning based on signals of opportunity,” *IEEE Communications Letters*, vol. 18, no. 2, Feb. 2014. 186
- [158] A. Nagy et al., “Time-based localisation in unsynchronized wireless lan for industrial automation systems,” in *IEEE Conf. on Emerging Technologies & Factory Automation (ETFA ’11)*, 2011. 186
- [159] X. Wu and Z. Gu, “A joint time synchronization and localization method without known clock parameters,” *Pervasive and Mobile Computing*, vol. Sep., 2016. 186, 190

REFERENCES

- [160] D. Spano and R. F., “A joint time synchronization and localization method without known clock parameters,” in press for *Pervasive and Mobile Computing*, 2016. 186
- [161] “Comparisons of error characteristics between toa and tdoa positioning,” *IEEE Trans. on Aerospace and Electronic Systems*, vol. 38, 2002. 187
- [162] K. Miller, “On the inverse of the sum of matrices,” *Mathematics Magazine*, vol. 54, no. 2, March 1981. 194
- [163] F. Gringoli and L. Nava, “Openfwf: Open firmware for wifi networks,” available at <http://www.ing.unibs.it/openfwf/>. 206
- [164] D. Berger, F. Gringoli, N. Facchi, I. Martinovic, and J. Schmitt, “Gaining insight on friendly jamming in a real-world ieee 802.11 network,” in *Proceedings of the 7th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, July 2014. 206
- [165] P. Salvador, V. Mancuso, P. Serrano, F. Gringoli, and A. Banchs, “Voipiggy: Analysis and implementation of a mechanism to boost capacity in ieee 802.11 wlans carrying voip traffic,” *IEEE Transactions on Mobile Computing*, vol. In print, 2014. 206
- [166] P. Salvador, L. Cominardi, F. Gringoli, and P. Serrano, “A first implementation and evaluation of the ieee 802.11aa group addressed transmission service,” *ACM SigComm Computer Communication Review*, vol. 44, no. 1, pp. 36–41, 2014. 206
- [167] J. Xiong, K. Sundaresan, and K. Jamieson, “Tonetrack: Leveraging frequency-agile radios for time-based indoor wireless localization,” in *ACM Mobicom 2015*, 2015. 209
- [168] N. A. Alrajeh, M. Bashir, and B. Shams, “Localization techniques in wireless sensor networks,” *Hindawi International Journal of Wireless Sensor Networks*, vol. 9, no. 6, 2013. 211