KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Technology

Energy-efficient and secure implementations for the IoT

Winderickx Jori

Supervisors:
prof. dr. ir. Nele Mentens
dr. ir. Dave Singelée

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Technology (PhD)

March 2020

Energy-efficient and secure implementations for the IoT

Winderickx JORI

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Technology (PhD)

March 2020

Preface

Right after I graduated, I was given the opportunity by prof. dr. ir. Nele Mentens to start a PhD at the research group of Embedded Systems and Security (ES&S). I like to think it was because of my satisfactory work during my visit in Finland for my master thesis. It was about developing a security architecture for a Internet of Things platform of the open source initiative Ell-i. At that time, my interest in IoT systems and security was ignited. The topic of this thesis was, therefore, right up my alley.

I continued developing and analysing security architectures for the IoT during my PhD studies. We focused primarily on the impact of security algorithms and protocols on the performance and energy consumption of these constrained devices. Thus, you will find practical solutions and analyses of existing challenges in this thesis. Consequently, I think researchers, practitioners, and students interested in the implementation of security architectures for low-power and interconnected devices can benefit from reading my thesis.

This work would not have been possible without the projects that I worked on at ES&S and which were funded by IWT Flanders, FWO and Interreg V-A Euregio Meuse-Rhine. Furthermore, I would like to thank my supervisor prof. dr. ir. Nele Mentens and co-supervisor dr. ir. Dave Singelée, without whom my research contributions would be non-existent. Next, I have had the pleasure to work with numerous bright coauthors and colleagues, which enabled our work to reach higher levels. However, I do not dare to enumerate them all. Finally, special thanks to the people most close to me, like my parents, friends, and my sweetheart Eline, who have supported me emotionally throughout the many ups and downs while pursuing a PhD.

Abstract

The IoT or Internet of Things is defined by, for example, the Internet Engineering Task Force (IETF) as the network of physical objects or "things" embedded with electronics, software, sensors, and connectivity to enable objects to exchange data with the manufacturer, operator and/or other connected devices. Since the creation of the concept in 1999, the IoT has gained a lot of popularity, and, it is still increasingly used in various environments. This is also the reason for the numerous challenges that characterise the IoT.

This dissertation focuses on three important challenges to provide IoT security: heterogeneity, performance, and foremost energy-efficiency. The IoT is a heterogeneous environment due to the variety of devices, network architectures and wireless communication technologies that are used. Furthermore, these IoT devices typically have to adhere to performance requirements while they have limited storage and computation capabilities due to the fact that they are battery-powered.

The contribution of this dissertation is four-fold. The first contribution consists of providing end-to-end security in a heterogeneous environment. For this purpose, the use of object security is analysed and applied in a proof-of-concept system. The second contribution is the analysis and optimisation of using coupons in constrained devices. This technique reduces the computation cost of generating digital signatures. The third contribution is the in-depth analysis of the energy consumption of security algorithms in terms of both computation and communication cost. This energy-security analysis is used to define an approach on how to optimise the duration of security sessions to reduce the energy impact of a session-based security protocol. Finally, the fourth contribution is a healthcare use case, providing end-to-end security using a public-key approach on the one hand and a symmetric-key approach on the other hand.

In summary, this dissertation describes the research outcomes of the in-depth and practical study of the aforementioned security techniques to provide energy-efficient, performance-aware end-to-end security in a heterogeneous IoT environment.

Beknopte samenvatting

Het IoT of Internet der Dingen wordt door bijvoorbeeld de Internet Engineering Task Force (IETF) gedefinieerd als het netwerk van fysieke objecten of “dingen” die ingebed zijn in elektronica, software, sensoren en connectiviteit om objecten in staat te stellen gegevens uit te wisselen met de fabrikant, operator en/of andere aangesloten apparaten. Sinds de creatie van het concept in 1999 heeft het IoT veel aan populariteit gewonnen en wordt het nog steeds in toenemende mate gebruikt in verschillende omgevingen. Dit is ook de reden voor de vele uitdagingen die het IoT kenmerken. Dit proefschrift richt zich op drie belangrijke uitdagingen omtrent de beveiliging van het IoT: heterogeniteit, performantie en vooral energie-efficiëntie. Het IoT is een heterogene omgeving door de verscheidenheid aan apparaten, netwerkarchitecturen en draadloze communicatietechnologieën die worden gebruikt. Bovendien moeten deze IoT-apparaten doorgaans voldoen aan performantie-eisen, terwijl ze slechts beperkte opslag- en rekenmogelijkheden hebben vanwege het feit dat ze typisch op batterijen werken.

De bijdrage van dit proefschrift is viervoudig. De eerste bijdrage bestaat uit het leveren van end-to-end beveiliging in een heterogene omgeving. Hiervoor wordt het gebruik van object security geanalyseerd en toegepast in een proof-of-concept systeem. De tweede bijdrage is de analyse en optimalisatie van het gebruik van coupons in apparaten met beperkte rekenkracht. Deze techniek verlaagt de berekeningskosten voor het genereren van digitale handtekeningen. De derde bijdrage is de diepgaande analyse van het energieverbruik van beveiligingsalgoritmen in termen van zowel berekenings- als communicatiekosten. Deze energie-beveiligings-analyse wordt gebruikt om een aanpak te definiëren voor het optimaliseren van de duur van veiligheidssessies om de energie-impact van een sessie-gebaseerd veiligheidsprotocol te verminderen. Ten slotte is de vierde bijdrage een use case in de gezondheidszorg, die end-to-end beveiliging voorziet door middel van een asymmetrische-sleutel benadering enerzijds en een symmetrische-sleutel benadering anderzijds.

Samengevat beschrijft dit proefschrift de onderzoeksresultaten van de diepgaande en praktische studie van de bovengenoemde beveiligingstechnieken om energie-efficiënte en performantiebewuste end-to-end beveiliging te voorzien in een heterogene IoT-omgeving.

List of Abbreviations

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
ABP	Activated By Personalisation
ADC	Analog-to-Digital Converter
AEAD	Authenticated Encryption with Associated Data
AKE	Authenticated Key Establishment
AppSKey	Applicaiton Session Key
BioZ	bio impedance
BLE	Bluetooth Low Energy
CBC	Cipher Block Chaining
CBOR	Concise Binary Object Representation
CCM	Couter with CBC-MAC
CoAP	Constrained Application Protocol
COSE	CBOR Object Signing and Encryption Protocol
CTR	Counter
DH	Diffie-Hellman key exchange
DoS	Denial-of-Serice
DTLS	Datagram Transport Layer Security
E	Energy
EC	Elliptic Curve
ECB	Electronic Codebook
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECG	electrocardiogram
EMR	Electronic Medical Record

FPort	Port Field
GCM	Galois/Counter Mode
HTTP	Hypertext Transfer Protocol
I	Electric current
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
KDF	Key Derivation Function
LoRa	Long Range modulation technique
LoRaWAN	Low Power, Wide Area networking protocol
LPWAN	Low-Power Wide-Area Network
MAC	Message Authentication Code
MCU	microcontroller unit
MIC	Message Integrity Code
MITM	man-in-the-middle
NwkSKey	Network Session Key
OOB	Out-Of-Band
OSCOAP	Object Security of CoAP
OSCORE	Object Security for Constrained RESTful Environments
OTAA	Over-The-Air Activated
P	Electric power
PIN	Personal Identification Number
PPG	photoplethysmogram
PRNG	Pseudo-Random Number Generator
PSK	Pre-Shared Key
R	Electric resistance
SI	International System of Units
TCP	Transmission Control Protocol

TLS	Transport Layer Security
TRNG	True Random Number Generator
TTP	Trusted Third Party
U	Electric potential difference
UDP	User Datagram Protocol
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

List of Symbols

Ω	The unit of electrical resistance
A	The SI unit of electrical current
Ah	A unit of electric charge
C	The SI unit of electric charge
J	The SI unit of energy
s	The SI unit of time
V	The SI unit of electric potential difference
W	The SI unit of power

Contents

List of Figures

List of Tables

Chapter 1

Introduction

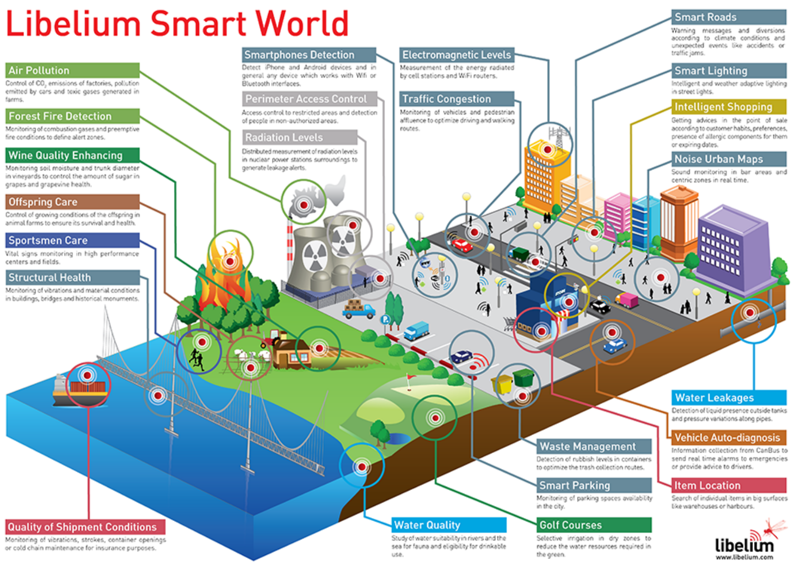
The term Internet of things (IoT) was likely introduced by Kevin Ashton that used the term in a presentation he gave at Procter & Gamble in 1999 [6]. Up till now, a globally accepted definition has not been established [23]. However, standardisation agencies like Internet Engineering Task Force (IETF), Institute of Electrical and Electronics Engineers (IEEE), and Internet of Things Global Standards Initiative (IoT-GSI) have made an effort to define the Internet of Things. In this thesis, we consider the following definition of IoT as defined by the IETF:

> *The Internet of Things is the network of physical objects or "things" embedded with electronics, software, sensors, and connectivity to enable objects to exchange data with the manufacturer, operator and/or other connected devices. – IETF*

The Internet of Things has since its adoption only seen more and more application domains like smart cities, Industry 4.0, and Internet of Medical Things [7, 23, 39, 129, 131, 132, 133]. An example of the range of possible application domains is imagined by Libelium in Figure 1.1.

1.1 Challenges in IoT

Billions of IoT devices have been and are being created and connected to the internet as analysed by GrowthEnabler [38], resulting in a billion-dollar market as shown in Figure 1.2. Note that the deployment of a vast number of devices



Air Pollution
Control of CO_2 emissions of factories, pollution emitted by cars and toxic gases generated in farms.

Forest Fire Detection
Monitoring of combustion gases and preemptive fire conditions to define alert zones.

Wine Quality Enhancing
Monitoring soil moisture and trunk diameter in vineyards to control the amount of sugar in grapes and grapevine health.

Offspring Care
Control of growing conditions of the offspring in animal farms to ensure its survival and health.

Sportsmen Care
Vital signs monitoring in high performance centers and fields.

Structural Health
Monitoring of vibrations and material conditions in buildings, bridges and historical monuments.

Smartphones Detection
Detect iPhone and Android devices and in general any device which works with Wifi or Bluetooth interfaces.

Perimeter Access Control
Access control to restricted areas and detection of people in non-authorized areas.

Radiation Levels
Distributed measurement of radiation levels in nuclear power stations surroundings to generate leakage alerts.

Electromagnetic Levels
Measurement of the energy radiated by cell stations and WiFi routers.

Traffic Congestion
Monitoring of vehicles and pedestrian affluence to optimize driving and walking routes.

Smart Roads
Warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.

Smart Lighting
Intelligent and weather adaptive lighting in street lights.

Intelligent Shopping
Getting advices in the point of sale according to customer habits, preferences, presence of allergic components for them or expiring dates.

Noise Urban Maps
Sound monitoring in bar areas and centric zones in real time.

Water Leakages
Detection of liquid presence outside tanks and pressure variations along pipes.

Vehicle Auto-diagnosis
Information collection from CanBus to send real time alarms to emergencies or provide advice to drivers.

Waste Management
Detection of rubbish levels in containers to optimize the trash collection routes.

Item Location
Search of individual items in big surfaces like warehouses or harbours.

Smart Parking
Monitoring of parking spaces availability in the city.

Golf Courses
Selective irrigation in dry zones to reduce the water resources required in the green.

Quality of Shipment Conditions
Monitoring of vibrations, strokes, container openings or cold chain maintenance for insurance purposes.

Water Quality
Study of water suitability in rivers and the sea for fauna and eligibility for drinkable use.

Figure 1.1: Libelium Smart World Infographic – Sensors for Smart Cities, Internet of Things and beyond [62].

would not be possible without affordable hardware, which is typically constrained in size, performance, etc. Furthermore, all these devices are operated without human intervention and are generating a large amount of information that is stored and processed in the cloud or used by other IoT devices.

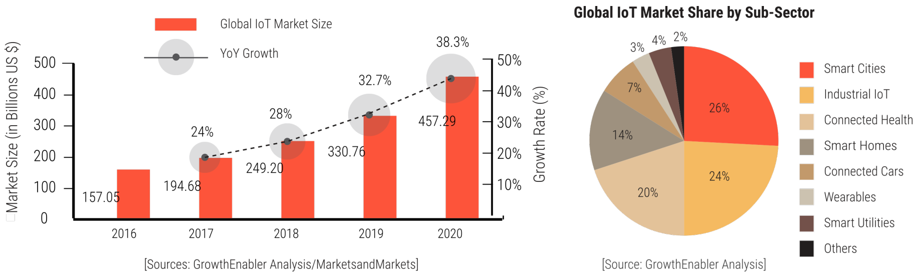


Figure 1.2: Global market size and growth forecast of IoT as analysed by GrowthEnabler in 2017 [38].

All these devices face the same and even more challenges than traditional and more powerful connected devices like personal computers or cloud servers. For example, vulnerabilities/applications have to be patched, certificates need to be updated, encryption keys need to be established, and/or data need to be communicated in real-time. The difference between an IoT device and a personal computer is that a human handler can address the issues of a personal computer locally when they arise, while an IoT device needs to address security issues autonomously or with limited human interaction, as it is typically deployed in a remote location. Thus, the management of billions of IoT devices requires strong authentication and authorisation protocols, since, it is done remotely via e.g. a cloud interface or a local IoT gateway.

Another challenging aspect of IoT systems is the amount of data that is generated by IoT devices. These data are usually generated in a remote location and may contain personal information related to e.g. industry processes or health metrics. Therefore, providing cryptographic properties like confidentiality and integrity protection is important. This is traditionally done using a public-key based key agreement protocol to establish an encryption key and a symmetric-key algorithm to encrypt the data. While these security protocols already exist for several years, many of them cannot be applied to the IoT. The foremost reasons are the large scale of IoT networks and the constraints on the hardware. For example, the protocols used in a certificate-based public-key infrastructure, which can provide strong authentication and key agreement, often require lots of performance, storage, and memory. An alternative to public-key based key agreement protocols is the use of pre-shared symmetric keys between an IoT node and a delegation server. Moreover, the node offloads the authentication and authorisation procedure to the delegation server. The downside of this method is that the shared keys have to be updated more frequently, because, they have a higher risk of compromise.

In summary, the development of a security architecture for IoT systems is a difficult challenge, considering lots of different requirements need to be taken into account, e.g. throughput, latency, security, and/or energy consumption. For this reason, further research is needed in the evaluation of how to apply security in IoT systems and in identifying the most optimal solutions.

The following challenges are considered in this thesis: (1) securing heterogeneous devices and networks, (2) enabling digital signatures on constrained IoT devices, (3) analysing the energy consumption of security protocols, and (4) providing security solutions for mobile health systems. These challenges are presented in Figure 1.3 and are described in more detail in the following four paragraphs.

An important challenge of the IoT is the heterogeneity of the connectivity and devices that are used to enable the plethora of applications that can be envisioned. For example, the 6LoWPAN network [69] is regularly used for smart homes, while, the LoRaWAN network [104] is designed for long-range and low-power applications like smart agriculture. Furthermore, networks can be

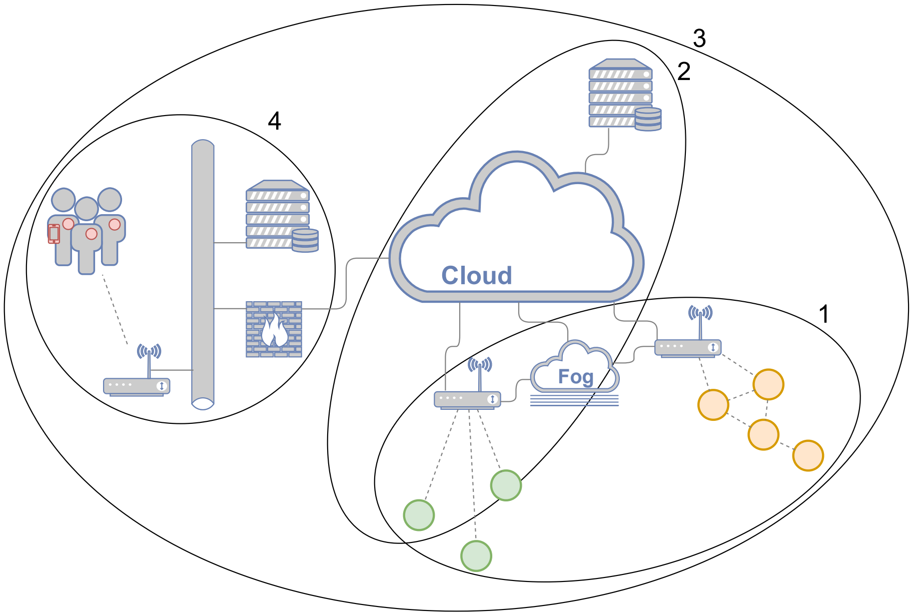


Figure 1.3: Graphical representation of the challenges tackled in this thesis (1: heterogeneity of devices and networks, 2: digital signatures for constrained IoT devices, 3: energy consumption of security, and 4: security architectures for an IoT system).

hosted by multiple parties which may render similar networks not interoperable. We propose to solve this issue by introducing an intermediate node that can communicate with both networks, without renouncing end-to-end security. The intermediate node is assumed to be more powerful than the end nodes and is typically referred to as the "fog". We refer to the proposed security architecture as "HeComm: Heterogeneous Communication". A prototype implementation of HeComm in a heterogeneous network consisting of 6LoWPAN nodes and LoRaWAN nodes is built on top of the Constrained Application Protocol (CoAP). The security properties and the memory utilisation of the constrained nodes are evaluated.

Another challenge of the IoT is to provide strong security properties to IoT systems, since, IoT nodes are typically constraint in one or more of the following features: storage (non-volatile memory), memory (volatile memory), energy/power, and/or performance. One solution that provides strong security properties like mutual authentication, confidentiality, and integrity protection is end-to-end security using the pre-shared key technique. It has a low impact on the previously mentioned constraining features, which has been demonstrated by a multitude of authors [46, 70, 85]. On the other hand, using public-key based cryptography can provide stronger cryptographic properties, but, is often avoided in IoT networks as it is much more computationally expensive. Lots of research has already been done to increase the efficiency of public-key based algorithms [41, 42, 63], though, some methods remained unexplored in an IoT setting [120, 123]. In this thesis, we explore the efficient implementation of public-key digital signatures and signcryption schemes in low-end embedded devices, focusing on the trade-off between computation and storage.

IoT nodes are typically battery-powered, as they are usually remotely positioned. This limits the energy budget of the application. Consequently, the impact of the security mechanisms on the node's lifetime should be reduced to a minimum as the application has the highest priority. Many studies have already analysed the energy consumption of security protocols [28, 46, 58, 99], however, a thorough analysis of the overall energy consumption, taking into account not only the computation energy, but also the wireless communication energy, is missing. The third focus of this thesis is to provide an in-depth analysis of the complete energy cost of security protocols deployed over a wireless communication link.

An example of a popular IoT application domain is mobile health, as discussed by the KU Leuven metaforum [22]. According to Claes et al., mobile health, which stands for the measuring of human activity, behaviour, etc. using mobile devices, "can potentially drag medicine from the clinic to the consumer, where users are able to read, monitor and manage their health information" [22]. Mobile health implies low-power systems, thus, a long lifetime of the mobile devices is important to facilitate their use. Furthermore, the data generated by these systems require strong cryptographic properties, since, they handle sensitive data. In this thesis, we evaluate two security architectures to provide energy-efficient and strong security to a wearable health sensor system.

Throughout the whole thesis, the focus is on the optimisation of cryptographic protocols and implementations that provide end-to-end security. An attack vector that we do not consider, is side-channel analysis. Side-channel analysis attacks aim at extracting secret information from a computing device through side channels such as the power consumption [53], the electromagnetic radiation [36] and the timing behaviour [54].

Privacy consists of different aspects, as discussed by Pfitzmann et al. [80], who proposed terminology for privacy measures like anonymity, unlinkability, unobservability, and pseudonymity. For example, the anonymity of a subject means that the subject is not identifiable within a set of subjects, while, the unobservability of an item of interest requires that an item cannot sufficiently be distinguished whether it exists or not. To prevent, for example, the identification of RFID devices by and adversary, a lot of research has gone into the design of privacy-preserving RFID authentication protocols [44, 61, 79]. However, a distinction can be made in terms of the adversary, as proposed by Peeters et al. [79], between a weak attacker who can eavesdrop on the communication channel, and a strong attacker who can e.g. read the memory of an RFID tag. In terms of solutions, researchers like Hermans et al. [44] showed that public-key cryptography is required to protect against a strong attacker by modelling privacy for RFID systems.

Privacy measures can be provided through various techniques. For example, privacy could be one of the design goals of an authentication protocol, such as the privacy-preserving RFID authentication protocols discussed in the previous paragraphs. But it can also be realised using other methods, such as the use of pseudonyms or mix networks. Pseudonyms are custom identifiers that are not linked to the devices' identifiers, while, mix networks is a technique that provides hard-to-trace communication channels within the network, among other by applying the technique of onion routing. To systematically identify and mitigate privacy threats of a system, the LINDDUN privacy threat modelling methodology [128] can be used. Our work does not systematically take into account privacy, since, we focus on providing end-to-end communication security. Thus, we do not focus on securing the system, i.e. to provide aspects like privacy and authorisation.

In the proposed solutions in this thesis, we will focus on a 128-bit security level. The sizes of the cryptographic keys are chosen to meet this requirement. They should provide sufficient security until 2028, as proposed by the Cryptographic Key Length Recommendation [37], Smart et al. [74], and ECRYPT-CSA [103]. Most use cases in an IoT environment like the applications considered in this thesis, only have a limited lifespan, i.e. a couple of years. If applications have a longer lifespan, a higher security level should be considered.

1.2 Outline

The thesis starts by introducing the necessary background concepts in Chapter 2. Then, the challenge of providing a security architecture in a heterogeneous environment is discussed and the HeComm architecture is proposed in Chapter 3. This is followed by the study to provide digital signatures and signcryption schemes in IoT systems, where we analyse the trade-offs between storage and computation requirements in Chapter 4. Next, an in-depth energy analysis of security protocols and algorithms in secure IoT systems is discussed in

Chapter 5. Then, in Chapter 6, two different security architectures are proposed for a wearable medical system, designed to monitor patients in a hospital. Finally, in Chapter 7, a conclusion is given which summarises the overall contributions and potential future work.

1.3 Publications

International journal submissions

2019 Winderickx, J., Bellier, P., Coppieters, D., Duflot, P., and Mentens, N. Communication and security trade-offs for battery-powered devices: a case study on wearable medical sensor systems. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* (2019), 17 pages. [submitted]

(Contributes to Chapter 6)

2019 Winderickx, J., Braeken, A., Singelée, D., and Mentens, N. In-depth energy analysis of security algorithms and protocols for the Internet of Things. *ACM Transactions on Internet of Things (TIOT)* (2019), 18 pages. [submitted]

(Contributes to Chapter 5)

2019 Winderickx, J., Braeken, A., and Mentens, N. Enhanced end-to-end security through symmetric-key cryptography in wearable medical sensor networks. *ACM Transactions on Computing for Healthcare (HEALTH)* (2019), 17 pages. [submitted]

(Contributes to Chapter 6)

International journal papers

2019 Rabbani, M. M., Vliegen, J., Winderickx, J., Conti, M., and Mentens, N. SHeLA: Scalable Heterogeneous Layered Attestation. *IEEE Internet of Things Journal* (2019), 12 pages

(Not included in this dissertation)

International conference papers

2016 Windericks, J., Daemen, J., and Mentens, N. Exploring the use of shift register lookup tables for Keccak implementations on Xilinx FPGAs. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)* (Lausanne, Switzerland, 2016), J. Anderson and P. Brisk, Eds., IEEE, pp. 454–457

(Not included in this dissertation)

2016 Winderickx, J., Daemen, J., and Mentens, N. On the parallelization of slice-based Keccak implementations on Xilinx FPGAs. In *6th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016)* (Barcelona, Spain, 2016), G. D. Natale and I. Polian, Eds., pp. 103–107

(Not included in this dissertation)

2017 Picek, S., Yang, B., Rozic, V., Vliegen, J., Winderickx, J., De Cnudde, T., and Mentens, N. Prngs for masking applications and their mapping to evolvable hardware. In *International Conference on Smart Card Research and Advanced Applications (CARDIS)* (Cannes, France, 2017), K. Lemke-Rust and M. Tunstall, Eds., Springer International Publishing, pp. 209–227

(Not included in this dissertation)

2018 Winderickx, J., Singelée, D., and Mentens, N. HeComm: End-to-end secured communication in a heterogeneous IoT environment via fog computing. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (Las Vegas, Nevada, USA, 2018), B. Lee and J. Kang, Eds., IEEE, pp. 1–6

(Contributes to Chapter 3)

2018 Winderickx, J., Braeken, A., Singelée, D., Peeters, R., Vandenryt, T., Thoelen, R., and Mentens, N. Digital signatures and signcryption schemes on embedded devices. In *Proceedings of the 15th ACM International Conference on Computing Frontiers - CF '18* (Ischia, Italy, 2018), M. Moretó and J. Weidendorfer, Eds., ACM Press, pp. 342–347

(Contributes to Chapter 4)

2018 Winderickx, J., Braeken, A., and Mentens, N. Storage and computation optimization of public-key schemes on embedded devices.

In *2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech)* (Brussels, Belgium, 2018), IEEE, pp. 1–8

(Contributes to Chapter 4)

2019 Winderickx, J., Bellier, P., Duflot, P., Coppieters, D., and Mentens, N. Work-in-Progress: Communication and security trade-offs for wearable medical sensor systems in hospitals. In *Proceedings of the International Conference on Embedded Software Companion - EMSOFT '19* (New York, New York, USA, 2019), S. Sankaranarayanan and T. Bourke, Eds., ACM Press, pp. 1–2

(Contributes to Chapter 6)

Chapter 2

Background

In this chapter, necessary background concepts are introduced. First, an overview of the most common wireless networks and their security architectures is given in Section 2.1. Then, we continue with an introduction on IoT platforms and the development of IoT applications in Section 2.2. Section 2.3 provides the reader with some background knowledge on cryptography. Next, a high-level overview of the protocols that are required to provide secure communication between two entities is given in Section 2.4. Finally, two methods on how to measure the energy consumption are discussed in Section 2.5.

2.1 Wireless networks

In an IoT environment many different kind of applications are possible e.g. wearable sensors, and remote weather stations. Typically, the IoT involves wireless communication and battery-powered devices. Different types of wireless networks have been and are being devised to accommodate the needs of these applications. In this thesis, three types are considered: Wireless Local Area Network (WLAN), Wireless Personal Area Network (WPAN), and Low-Power Wide-Area Network (LPWAN). They can be distinguished by multiple parameters like range, throughput, energy consumption, and security. For example, the expected range of these types of networks is presented in Figure 2.1. In this thesis, we will focus primarily on the security and the energy consumption aspect. The firmware cost in terms of storage and memory can be calculated in advance in most cases, while, the amount of operations or interactions during operation are harder to predict as they depend on the availability, popularity of the service, etc.

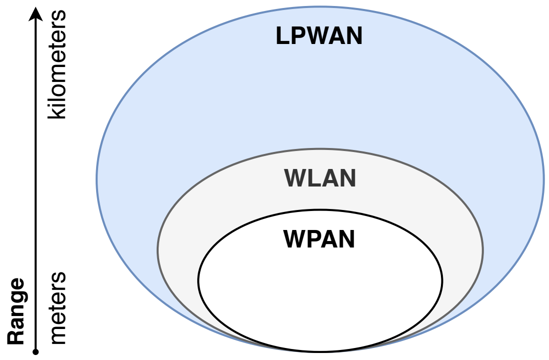


Figure 2.1: Indication of the expected range of the considered wireless network types.

WPANs are typically used in the personal area of a person or entity, e.g. a home or a factory. An example application is smart lighting. In this setting, the lights of a room or building can be controlled via a smartphone. Networks that classify as WPAN are ZigBee, 6LoWPAN, and Bluetooth Low Energy (BLE).

Another example of an application is patient surveillance. The health of a patient can wirelessly be measured and monitored in real-time by the medical staff. It can be used to avoid the clutter of cable management when e.g. a patient enters the hospital to be monitored. While, WPAN networks could also be used to implement this system, WLAN networks like Wi-Fi may be more suitable. They support higher data throughput, and, the Wi-Fi infrastructure is typically already available.

Internet of Things applications are not limited to indoor applications, e.g. smart cities and smart agriculture. The battery-powered devices in these types of applications are typically put in remote locations with lifetime requirements of multiple years. For this purpose, the Low-Power Wide-Area Network (LPWAN) classification exists. In this thesis, the LoRaWAN standard is considered in this category.

2.1.1 Wi-Fi

Wi-Fi is a wireless network protocol based on the IEEE 802.11 standard [1], introduced by the Wi-Fi Alliance in the year 2000 [117]. The IEEE 802.11

standard was introduced by the Institute of Electrical and Electronics Engineers (IEEE) in 1997 and has received since its conception multiple revisions and amendments. This thesis will focus on the most prevalent version of Wi-Fi at this moment, namely 802.11n. It is based on the IEEE 802.11n-2009 amendment, and, it is secured via the WPA2 protocol. WPA2 is described in the IEEE 802.11i-2004 amendment.

The 802.11n standard specifies a physical (PHY) and a Medium Access Control (MAC) layer. The PHY layer deals with the physical world, i.e. it uses telecommunication techniques like direct-sequence spread spectrum (DSSS) and Orthogonal Frequency Division Multiplexing (OFDM) to communicate messages. The MAC layer provides reliable data transfer between entities, e.g. channel access and link addresses. Moreover, it consists of a set of management functions, and, it defines the structure of the message.

The WPA2 protocol provides secure data transfer between links. Its name is the trademark used by the Wi-Fi Alliance to describe the IEEE 802.11i-2004 amendment. It superseded the WEP and WPA protocol after serious security weaknesses were found by researchers [45]. Furthermore, a newer version WPA3 has also been devised. However, it has not seen widespread adoption yet. Generally, two configurations of the WPA2 protocol are used for the authentication of entities and the key distribution: WPA-Personal and WPA-Enterprise. WPA-Personal defines the use of a Pre-Shared Key (PSK), i.e. both entities share the same password. WPA-Enterprise identifies the use of the IEEE 802.1X standard. This standard uses the Extensible Authentication Protocol (EAP) and its many configurations to provide a range of security levels. This configuration is often only used in strictly managed environments like factories, as it requires an authorisation server, which controls the access of entities to the wireless network.

2.1.2 BLE

Bluetooth is an open standard maintained by the Bluetooth SIG community [10]. It can be used to create a wireless personal area network. Since its creation, multiple versions of Bluetooth have been devised. The most notable are the following: Basic Rate (BR), Enhanced Data Rate (EDR), and Low Energy (LE). BR and EDR are designed for continuous data streaming and support a point-to-point and a piconet network topology. A piconet topology consists of one master device that serves multiple slave devices. For BR/EDR networks, a piconet can have up to 7 active and 255 inactive slave nodes. Additionally, each slave could potentially create its own piconet to make a topology called scatternet. BR/EDR networks are typically used in applications like audio streaming. They were initially designed for low power scenarios, but, LE uses short burst data transmission to reduce the power consumption even further. In terms of network configurations, it supports besides the point-to-point and piconet topology, also, a mesh and a broadcast topology. Note that a piconet in Bluetooth Low Energy can have an unlimited amount of slaves.

In this section, we continue with a brief introduction on the security architecture of Bluetooth. For more information, please refer to the core specification of Bluetooth or the Guide to Bluetooth by NIST [77]. The Bluetooth standard specifies five basic security services: authentication to verify the identity, confidentiality to protect data compromise, authorisation to control resource access, message integrity to protect the transmitted data, and Pairing/Bonding to create shared secret keys [77]. In terms of security, the BR and EDR specification specify four security modes. Mode 1 uses no security. Mode 2 and 4 are service level procedures meaning that a link is established before initiating security services. Mode 3 is a link level procedure, i.e. security services are initiated before a link is established. Mode 2 uses a local security manager that controls access to specific application services. In mode 4, Bluetooth services need to specify their security requirements using one of five levels, which range from (level 0) no security to (level 4) Authenticated link key using Secure Connections required. NIST recommends the use of mode 4 and optionally mode 3 if mode 4 is not supported. Bluetooth Low Energy security modes' names are similar to BR/EDR modes since Bluetooth version 4.1, but, they differ slightly. For example, LE specifies that each service request can have its own security requirements.

The **Pairing service** provides the generation of a secret symmetric key. It is called the Link Key in Bluetooth BR and EDR and the Long Term Key in Bluetooth Low Energy. In Bluetooth BR and EDR, security modes 2 and 3 can perform the pairing service via either the Personal Identification Number (PIN) Pairing or Secure Simple Pairing (SSP) protocol, and, security mode 4 can only use SSP. SSP uses the Elliptic Curve Diffie-Hellman (ECDH) algorithm to establish a shared key. Furthermore, it has the four following association models to generate an initial Temporary Key (TK): Numeric Comparison, Passkey Entry, Just Works, and Out-Of-Band (OOB). Numeric Comparison can be used if a six-digit number can be displayed on both devices, and, if the user can verify the number on both devices. Passkey Entry requires at least one device that supports a keyboard interface so that the user can input a provided/generated secret into the device(s). If an OOB channel, i.e. a communication channel besides Bluetooth, is present then the key can be transported from one device to the other. Finally, if none of the other methods are available, Just Works is used. In this mode, the TK is set to all zeros. Since Bluetooth version 4.2, LE has the Secure Connection Key Generation which uses also the ECDH protocol to generate the long term key.

The generated key is used to provide the **confidentiality** and **integrity** service. In Bluetooth LE the AES-CCM algorithm is used, while, Bluetooth BR/EDR can use either the E0 stream cipher or the AES-CCM algorithm. However, the use of E0 is not recommended by NIST, as it is not a FIPS-approved algorithm. To **authenticate** the devices and the used keys, a challenge-response scheme is used. Note that the knowledge of the long term key is implied through its use in Bluetooth LE. In BR and EDR, the authentication service uses either the Legacy Authentication scheme or Secure Authentication scheme. The legacy scheme is based on the E1 algorithm, while, the secure scheme uses the HMAC-SHA algorithm. HMAC-SHA can generate a Message Authentication Code (MAC) on a message using the hash algorithm called SHA. The **authorisation** service is implemented using different levels of trust and customisable policies regarding service security.

2.1.3 6LoWPAN

6LoWPAN [69] is a specification to enable IPv6 communication over Low-Rate WPANs (LR-WPAN), specially IEEE 802.15.4 [43], as initially conceptualised by Mulligan [71]. It defines encapsulation and header compression techniques to reduce the size of an IPv6 packet. Since its use, 6LoWPAN is typically used on top of IEEE 802.15.4, which is a wireless network standard that specifies the physical and the mac layer on the OSI model. This wireless network defines two types of connected nodes: Full Function Devices (FFD) and Reduced Function Devices (RFD). It can operate in either a peer-to-peer/mesh or a star topology, and, it is controlled by a network coordinator which is a FFD that is generally mains powered. The RFD is a node that cannot act as a coordinator, i.e. a low power sensor node. In terms of security, IEEE 802.15.4 provides the following security services, depending on the chosen security level: data confidentiality, data authenticity, and replay protection. The security level ranges from (level 0) no security to (level 7) 128-bit AES-CCM data confidentiality and integrity protection using a 16 byte Message Integrity Code (MIC). In the header of a IEEE 802.15.4 packet, a field is provided to identify the used encryption key.

The 6LoWPAN specification does not provide the establishment and the maintenance of the encryption key. However, it is provided by other wireless network standards like Zigbee IP, which have been built on top of the IEEE 802.15.4 and the 6LoWPAN specification.

2.1.4 LoRaWAN

Low Power, Wide Area networking protocol (LoRaWAN) is the specification maintained by the LoRA Alliance [104] for a Low-Power Wide-Area Network. It is designed for wirelessly connected objects and tries to answer the following challenges of the IoT: bi-directional communication, end-to-end security, mobility, and localisation services. It uses a star-of-stars topology, as shown in Figure 2.2. Devices or IoT nodes connect wirelessly to gateways which convert LoRa packets to IP packets and vice versa. The gateway sends and receives data to and from the Network server that manages the LoRaWAN network. The Network server in its turn relays the data packets to and from the respective application server.

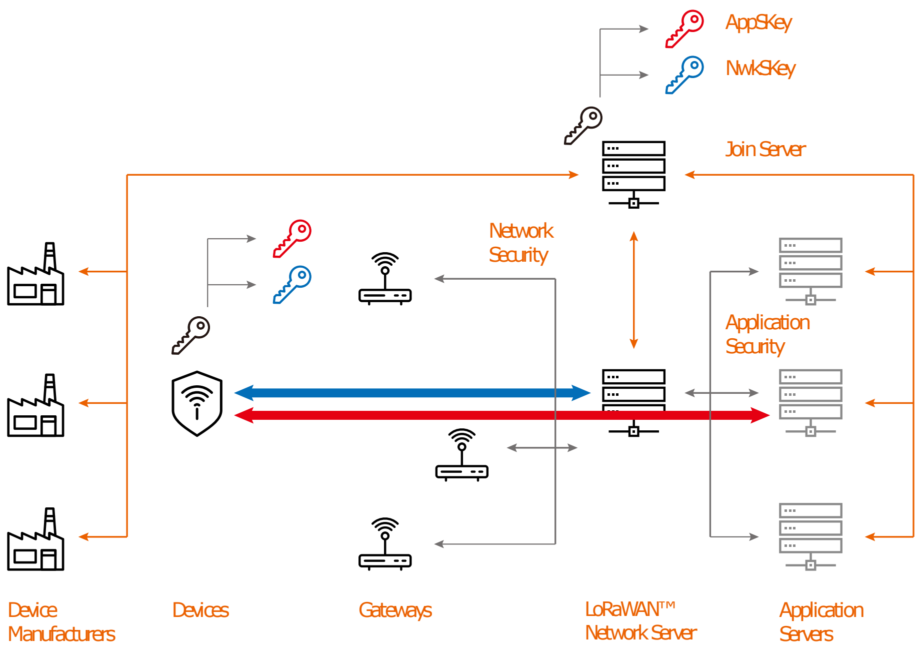


Figure 2.2: The LoRaWAN architecture as envisioned by the LoRa Alliance [64].

The data flow in a LoRaWAN network is protected by a set of two keys, i.e. AppSKey and NwkSKey. The payload/data of a packet is encrypted using the AES algorithm in CTR mode to provide confidentiality protection from the IoT node until the Application server. Next, a Message Integrity Code (MIC) is computed on the encrypted message via the network session key (NwkSKey) using the AES-CMAC algorithm [48] to protect the authenticity and integrity of the message. However, the MIC is already verified or computed at the

Network server. Thus, it does not provide end-to-end authenticity and integrity protection between the IoT node and the Application server. All keys that are used in the LoRaWAN network have a length of 128 bits.

The two following methods can be used to initialise the set of session keys: Activated By Personalisation (ABP), or Over-The-Air Activated (OTAA). In ABP mode, the session keys are preconfigured on the device. In OTAA mode, they are generated during the join procedure in OTAA mode. At the device manufacturers, each device is provisioned with a Application key (AppKey) and a globally unique identifier (DevEUI). Likewise, LoRaWAN networks have a 24-bit unique identifier assigned by the LoRa Alliance. The AppKey is only used in OTAA mode, as it is used to generate the session keys. They are computed via the AES-ECB encryption of nonces and identifiers transmitted and received during the join procedure, as shown in Equation (2.1). The join procedure consists of two messages: join request, and join accept. The IoT node sends a join request, which contains the Application identifier (AppID), DevUID, and a nonce (DevNonce), and, it is protected using a MIC generated using the AppKey. The Join Server will reply with a join accept if the request is successfully validated. This reply message contains a nonce (AppNonce), network identifier (NetID), etc. The join accept is protected using the MIC, and, it is encrypted using AES-ECB in decryption mode. Note that the join procedure is designed so that an IoT node only requires the AES encryption mode. The generated session keys are then distributed by the Join server to the respective Network and Application server. See the specification of the LoRaWAN network [104] for more information.

$$\begin{aligned} \text{NwkSKey} &= \text{AES}_{encrypt}(\text{AppKey}, \texttt{0x01}|\text{AppNonce}|\text{NetID}|\text{DevNonce}|\text{pad}_{16}) \\ \text{AppSKey} &= \text{AES}_{encrypt}(\text{AppKey}, \texttt{0x02}|\text{AppNonce}|\text{NetID}|\text{DevNonce}|\text{pad}_{16}) \end{aligned} \tag{2.1}$$

2.2 IoT platforms

IoT platforms mostly consist of the following five major components on a Printed Circuit Board (PCB): a physical interface (PHY), a microcontroller unit (MCU), sensors (S), a power management unit (Power), and a wireless interface (RF). They are wired to each other to provide power and communication, as shown in Figure 2.3. The most vital parts of a platform are the power management and the MCU component. The power management provides the correct voltage to all other components and manages the battery (charge and discharge) if it is present on the platform. The MCU, on the other hand, controls all components on the board to e.g. configure or readout a component. At the very least, it features a processor (to run the program), memory (to store volatile data), storage (to store non-volatile data), and physical interfaces (to communicate with other components). The PHY enables external wired communication to the board, e.g. USB, JTAG, and SWD. Similarly, the RF provides external wireless communication. The difference is that the PHY is mostly used for debugging or configuration, while, the RF is often used as the primary source of communication. The final component is the sensor that provides the digital representation of a physical property. For example, the Analog-to-Digital Converter (ADC), which is typically provided in the MCU, can be used to measure the voltage on an analog pin.

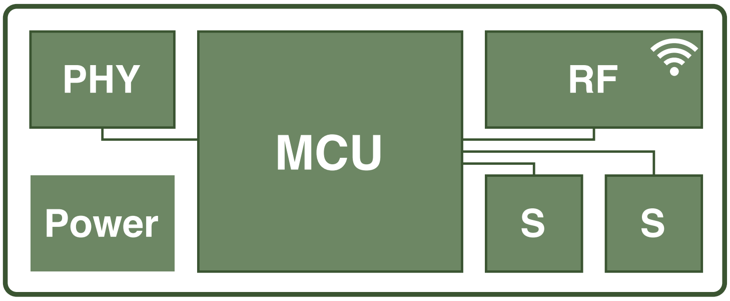


Figure 2.3: Schematic representation of an IoT platform and its essential components (PHY: physical interface, MCU: microcontroller unit, S: sensors, Power: power management unit, and RF: wireless interface).

An application is defined by the program that is stored in binary format on the MCU. It contains the instructions and data content that provide the functionality of the program for the respective MCU's architecture. The architecture of the MCU determines which instructions are available to the program. To improve the readability of the program in binary format, it could be represented using the Assembly language, e.g. ARMv7-M Architecture [3]. However, writing a program using the Assembly language, i.e. per instruction, is very tedious. Therefore, the C or C++ programming language is typically used. It uses a compiler to convert the written C/C++ code to the required binary format. For example, the GNU Arm Embedded Toolchain [5] contains tools to compile and analyse the binary code.

The C/C++ compiler and its configuration greatly influences the performance and size of a program. For example, the GNU compiler has an option to control the optimisation level of the compilation process. Using the command line interface, the *-O* option can be passed with a range of additional specifications to specify the optimisation level. For example, *-O1...3* determines the overall level of optimisation and *-Os* tries to reduce the size of the program as much as possible.

Software libraries and Operating Systems (OS) can be used to facilitate the code writing process. Writing code without an OS is called bare-metal programming, since, no high-level abstractions are used. This means that everything has to be written or included manually, e.g. system component drivers or process parallelization support. An OS, on the other hand, typically contains some level of support for IoT platforms, i.e. drivers and etc, to facilitate the programming process. Examples of operating systems that can be used for the types of embedded systems we use are the following: Mbed OS, RIOT-OS, Contiki-OS, and FreeRTOS. However, an OS might also provide too much overhead in terms of performance or required resources in some situations.

2.3 Cryptography

Cryptography is a means of providing information security. The following definition of cryptography is given in the handbook of applied cryptography by Menezes et al. [67]: "*Cryptography* is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication." These listed aspects are also referred to as cryptographic goals or properties. The following four basic properties are often used:

Confidentiality or privacy protects information from illegitimate access to data. Only those who have been authorised should be able to gain access to the data.

Data integrity detects unauthorised manipulation of data, e.g. insertion, deletion, and substitution of bits.

Authentication provides identification of the information or entities. To illustrate, two communicating parties should be able to identify each other and the origin of the communicated data.

Non-repudiation prevents an entity from denying previous commitments or actions.

The goal of *cryptography* is to address one or more of these four properties to some extent in both theory and practice. On which level we focus on each property depends on the application and the resources available. In our scenarios, we have to limit the performance and the amount of resources to provide security, since, we only consider constrained battery-powered devices. Furthermore, each application has a different setting and, therefore, different requirements in terms of security level. The security level is generally quantified as the amount of work required by an adversary to defeat an objective, e.g. to compromise the encryption key. To define the security requirements, one often uses the technique called *threat modelling*.

Threat modelling is a technique where we analyse the system to identify the goals and methods of an adversary. It is typically a thought experiment that should be done as early as possible in the design phase of an application to reduce the risk of major changes. A structural approach to threat modelling is the STRIDE technique. For more information on how to use the STRIDE technique or threat modelling, we refer to the book written by Adam Shostack called “Threat Modeling” [101]. STRIDE is an acronym that stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. It focuses on identifying the threats on the systems, i.e. enumerating the things that might go wrong. By identifying the threats, mitigations can be devised and designed into the application. For example, data transmitted via a wireless communication channel is at risk of exposure, thus, confidentiality protection is required.

Cryptographic primitives or tools are used to provide the required security properties of an application. Many primitives have already been devised in the literature, but, we will focus only on the primitives used in this thesis. The following five cryptographic tools are described: unkeyed primitives, symmetric-key primitives, random number generators, and key derivation functions.

2.3.1 Unkeyed primitives

Unkeyed primitives require only one input parameter (a message), e.g. cryptographic hash functions. They take a massage of arbitrary length and output a fixed-length message, i.e. a compact representation of the input (message digest). While collisions (multiple input that are mapped to the same output) are unavoidable, the output is typically sufficiently large that it is improbable to find such collisions. Hash functions are often used to provide data integrity. In this thesis, we will focus on the following two hash functions: SHA256 and SHA3-256. The suffix of these functions describe the length of the message digest, 256 bit in this case.

SHA256 is a hash function that belongs to the SHA-2 family of hash algorithms published by NIST in FIPS180-4 [27]. It has two stages: pre-processing and hash computation. First, the input message is parsed and padded into block-size blocks of data. SHA256 has a block size of 512 bit. Secondly, a message schedule is generated from the padded message. This schedule consists of functions, constants, and word operations that are used to generate series of hash values. The final hash value determines the message digest. In total, six logical functions and 64 predefined 32-bit words are used to compute the hash.

The SHA-2 family was designed behind closed doors at NSA and published in 2001. SHA-3, on the other hand, was the result of an open call of NIST for novel hash function proposals, where every candidate algorithm provided a description, design rationale and preliminary cryptanalysis [34]. The winning algorithm was submitted by Bertoni et al. [9] as a hash function called Keccak. It uses a sponge construction and internally the Keccak-f permutation. The Keccak-f permutation was designed to use only transposition, bit-level addition, and multiplication operations. Furthermore, these operations were arranged specifically to allow efficient software implementations. The advantage of SHA-3 over SHA-2 is the strong security rationale behind it.

2.3.2 Symmetric-key primitives

Symmetric-key primitives require two input parameters: a key and a message. For example, one key (secret key) is used by the symmetric-key algorithm to encrypt and decrypt the data. Furthermore, these primitives can also be used to create a Message Authentication Code (MAC), which provides data authentication. Two classes of these primitives are generally used: block ciphers and stream ciphers. Block ciphers break up the plaintext messages to fixed-size blocks which are processed individually, while, stream ciphers can process one symbol at a time, i.e. block length of one. The AES algorithm is a block cipher and is often used as a symmetric-key primitive. It is a standard published by NIST in FIPS 197 [75], and, it was originally designed by Daemen et al. as the Rijndael algorithm [26]. The input, state, and output have a length of 128 Bits. The key that is used can be 128, 192, or 256 Bits. Note that the key size defines the number of rounds executed, respectively 10, 12, and 14 rounds. The AES algorithm uses a round function that has four different byte-level operations: substitute using a substitution table (S-box), shift rows, mix columns, and add round key. For more information, we refer to the AES standard specification [75].

Block ciphers use certain block cipher modes to encrypt/decrypt data. The following modes are often used: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Counter (CTR), Couter with CBC-MAC (CCM), and Galois/Counter Mode (GCM). ECB, CBC, and CTR mode were part of recommendations of NIST published in 2001 [32]. Any given plaintext block gets encrypted to the same ciphertext block under a given key in ECB mode. It is, therefore, rarely used. CBC and CTR, on the other hand, use an Initialisation Vector (IV) to provide freshness that reduces the risk of reoccurring ciphertexts.

These modes can be used to either provide confidentiality or integrity. Integrity is provided when a Message Authentication Code (MAC) is computed on the message. To provide both confidentiality and integrity, Authenticated Encryption with Associated Data (AEAD) block modes like CCM and GCM were devised. CCM is described by Whiting et al. [116] in the standards proposal RFC3610 of IETF. GCM is proposed in FIPS 800-38D published by NIST [33].

2.3.3 Public-key primitives

A set of two keys are used in public-key primitives: a private and a public key. The keys can be used to provide digital signatures, signcryption, and key agreement. In short, public-key cryptography is based on the intractability of a mathematical problem. It means in practice that it should be infeasible for an adversary to calculate the private key via the public key. Thus, the public key may be known by all, and, the private key should remain a secret of an entity. This concept was introduced by Diffie and Hellman [29]. They proposed the Diffie-Hellman key exchange (DH) that establishes a shared secret between two entities over an unsecured channel. DH uses the discrete logarithm problem as the basis of its security. Basically, the public keys of both parties are shared to each other, and, a shared secret can be computed using their own private key and the public key of the other party. Without knowledge of one of the two private keys, it is computationally infeasible for an adversary to calculate the shared secret.

Another concept that uses public-key cryptography is digital signatures. It provides security properties like non-repudiation and data origin authentication, i.e. it is a means to bind information to an entity. Data origin authentication ensures a data packet to be originating from the entity that created the signature. Note that the digital signature concept does not provide confidentiality guarantees. This property is additionally provided in signcryption schemes. The use of these two types of schemes in an IoT setting are studied in Section 4. Moreover, the coupon technique, which is a performance optimisation technique for certain digital signature and signcryption schemes, is implemented and discussed. Consequently, a detailed description is given below for the Schnorr scheme and the signcryption scheme of Braeken et al. [14] that are respectively used as digital signature and signcryption schemes. We start with a description of the cryptographic operations used in the Schnorr and signcryption scheme. Then, the Schnorr and signcryption algorithm are explained. Finally, the comb method, which is the reference performance optimisation technique used in Section 4, is described.

Basic cryptographic operations

In order to execute the signature process and signcryption, operations based on Elliptic Curve Cryptography (ECC) are used. ECC was introduced by Miller and Koblitz in [68] and [52], respectively, and offers lightweight public-key cryptographic solutions. The algebraic structure of an Elliptic Curve (EC) over a finite field is exploited in ECC. An EC in the finite field F_p is defined by the equation $y^2 = x^3 + ax + b$ and denoted by $E_{p(a,b)}$, with a and b two constants in F_p and is required to be nonsingular which is defined by the equation $\Delta = 4a^3 + 27b^2 \neq 0$. We denote by P the base point generator of the EC of prime order q. All points on $E_{p(a,b)}$, together with the point at infinity form an additive group G. The group operation is called EC point addition. Cryptographic algorithms and protocols mainly use EC point multiplications, which consist of consecutive point additions.

Besides EC operations, other basic cryptographic algorithms are used in the digital signature and the signcryption schemes. In particular, there is the need for a one-way cryptographic hash function H compressing the data to an output in F_q^*. We will utilize the standard SHA-256. For symmetric key encryption, the standard AES is used, where both ECB and CTR mode are considered when applying the encryption for storage purposes. The AES-CCM mode is used in the signcryption scheme in order to also guarantee the integrity. We denote $c = E_k(m)$ and $m = D_k(c)$ as the encryption and decryption process of the message and ciphertext, respectively, using the key k. Furthermore, the concatenation of two messages M_1 and M_2 is denoted by $M_1 \| M_2$.

Signature and signcryption scheme

We assume a setting in which a constrained IoT device signs messages through the EC-based Schnorr scheme [96] or signs and encrypts messages through the EC-based signcryption scheme described in [14]. Therefore, we refer to the IoT device as the sender and to the remote party as the receiver. We assume that the IoT device always communicates with the same remote party and that the number of times it generates a signature or performs a signcryption can be determined before deployment.

EC-based Schnorr The EC-based Schnorr Signature scheme (EC-Schnorr) is chosen as it allows the sender to preprocess the most compute-intensive operation, being the EC point multiplication. This preprocess is also called the **Coupon technique**. We now briefly explain the coupon-based algorithm.

In the registration phase, sender and receiver agree upon an EC over F_p^* and a corresponding generator P of order q. Next, the sender obtains a key pair $(d_S, P_S = d_S P)$, consisting of private key and public key, respectively, e.g. through the Elliptic Curve Qu Vanstone (ECQV) certificate mechanism [18]. The sender is now able to produce a signature on the message m as follows:

- First, a random value $y \in F_q^*$ is chosen and $Y = yP$ is computed. The 2-tuple (y and Y) is referred to as a coupon in this paper; a number of Schnorr coupons are precomputed and stored on the sender's device, as proposed in [123].
- Next, the hash $h = H(m \| Y)$ is generated.
- The parameter, $d = y - hd_S$, is defined.

The 3-tuple (m, d, h) is sent to the receiver together with the identity and certificate of the signer. The signature can be verified by the receiver through the following steps:

- First, the public key of the sender P_S must be obtained taking the identity and certificate of the sender as input, e.g. by using the ECQV certificate mechanism.
- Next, the receiver computes $Y' = dP + hP_S$.
- Then, the hash $h' = H(m \| Y')$ is generated.
- Finally, the signature is accepted if the generated hash h' equals the received hash h.

EC-based signcryption In the EC-based signcryption scheme, encryption, denoted by $E_k()$, is added to the EC-Schnorr signature process. When the receiver is fixed, e.g. in the case when the sensor (sender) is always sending data to the same cloud platform, the most compute-intensive operations can again be preprocessed.

The registration phase is the same as in the EC-Schnorr scheme. Sender and receiver agree upon an EC over F_p^* and generator P of order q. Both the sender and the receiver obtain a key pair, (d_S, P_S) and (d_R, P_R), respectively.

We assume that the public key of the receiver is stored at sender's side. The following steps are then executed by the sender.

- First, a random value y is chosen and the points $Y = yP, k = yP_R$ are computed. The 3-tuple (y, Y, k) is referred to as a coupon throughout the paper; a number of signcryption coupons are precomputed and stored on the sender's device, as proposed in [123].
- The ciphertext is defined as $c = E_k(m)$.
- The hash $h = H(m\|Y\|ID_S\|P_S\|P_R)$ is computed.
- The parameter, $d = y - hd_S$, is defined.
- The output of the signcryption algorithm equals the 3-tuple (h, c, d), together with the identity and certificate of the sender.

Upon arrival of the signcrypted message (h, c, d), the receiver will perform the following steps to derive the original message m and to check the corresponding signature.

- First, the public key of the sender P_S is retrieved.
- Next, the receiver computes $Y' = dP + hP_S$.
- Then, the key $k' = d_R Y'$ is derived and thus $m' = E_{k'}(c)$.
- In the verification of the signature, the hash $H(m'\|Y'\|ID_S\|P_S\|P_R)$ should be equal to the received h. If this statement is correct, $m = m'$, if not, the output equals $\perp$.

Comb method

The point multiplication kP using the comb method optimisation considers a set of precomputed EC points representing an EC point (P) and a bit-string representation of a scalar (k). We will use and discuss the comb method as proposed by Hedabou et al. [42] in this thesis. The algorithm to represent the scalar as a bit-string is given in Figure 2.4. In short, the scalar k of length l is converted into d tuples, where $d = \lceil \frac{l}{w} \rceil$. Each tuple contains a bit-string ($\mathbb{K}_i = [K_i^{w-1}, \ldots, K_i^1, K_i^0]$) and a sign bit ($s_i = \pm 1$). Note that $k = K^{w-1}\| \ldots \|K^1\|K^0$ where each K^j is a bit-string of length d. And, K_i^j denotes the ith bit of bit-string j.

Computation of the bit-string representation of k

```
Input: an odd positive integer $k = (k_0, k_1, \ldots, k_{l-1})$, and a window width $w \geq 2$.
Output: the sequence of bit-strings $(\mathbb{K}_0, s_0), (\mathbb{K}_1, s_1), \ldots, (\mathbb{K}_{d-1}, s_{d-1})$.
1 :  $d = \lceil \frac{l}{w} \rceil$
2 :  For $i = l$ to $(wd - 1)$ do $k_i \leftarrow 0$.
3 :  $(\mathbb{K}_0, s_0) \leftarrow ([K_0^{w-1}, \ldots, K_0^1, K_0^0], 1)$.
4 :  For $i = 1$ to $i = d - 1$ do
5 :    If $[K_i^{w-1}, \ldots, K_i^1, K_i^0] \neq 0$ then $c_i \leftarrow 1$ else $c_i \leftarrow 0$.
6 :    $b[0] \leftarrow [K_{i-1}^{w-1}, \ldots, K_{i-1}^1, K_{i-1}^0], b[1] \leftarrow [K_i^{w-1}, \ldots, K_i^1, K_i^0]$.
7 :    $s[0] \leftarrow -1, s[1] \leftarrow 1$.
8 :    $(\mathbb{K}_i, s_i) \leftarrow (b[c_i], 1, (\mathbb{K}_{i-1}, s_{i-1}) \leftarrow (b[0], s[c_i])$.
9 :  Return $(\mathbb{K}_0, s_0), (\mathbb{K}_1, s_1), \ldots, (\mathbb{K}_{d-1}, s_{d-1})$.
```

Figure 2.4: Algorithm to calculate the bit-string representation of k [42].

Computation of kP using the bit-string representation of k

```
Input: a scalar k and an elliptic curve point $P$.
Output: $Q = kP$.
1 :  (Precomputation) Compute $[b_{w-1}, \ldots, b_1, b_0]P$ for all $(b_{w-1}, \ldots, b_1, b_0) \in \mathbb{Z}_2^w$.
2 :  If $k \bmod 2 = 0$ then $k' \leftarrow k + 1$ else $k' \leftarrow k + 2$.
3 :  Compute the sequence of bit-strings $(\mathbb{K}'_0, s_0), (\mathbb{K}'_1, s_1), \ldots, (\mathbb{K}'_{d-1}, s_{d-1})$ representing $k'$.
4 :  $Q \leftarrow \mathbb{K}'_{d-1}P$. //(each $\mathbb{K}'_j P$ will be fetched from the precomputed table)
5 :  for $i = d - 2$ down to $0$ do
6 :    $Q_1 \leftarrow 2Q$
7 :    $Q \leftarrow Q_1 + s_i \mathbb{K}'_i P$
8 :  $P' \leftarrow 2P$.
9 :  If $k \bmod 2 = 0$ then return $Q - P$ else return $Q - P'$.
```

Figure 2.5: Algorithm to calculate the point multiplication of kP using the comb method optimisation [42].

The bit-string representation of the scalar is used in the point multiplication algorithm presented in Figure 2.5. A set of EC points (Window) that represent P are precomputed $[b_{w-1}, \ldots, b_1, b_0]P$ for all $(b_{w-1}, \ldots, b_1, b_0) \in \mathbb{Z}_2^w$. This multiplication algorithm uses an uniform behaviour, by computing an adding and doubling operation at each step, to protect against a Simple Power Analysis (SPA) attack. An SPA attack is a type of physical attack where the adversary can extract the secret key by examining the information leaked by the device via e.g. time or power consumption.

2.3.4 Random number generators

Many cryptographic primitives depend upon the generation of unpredictable/random values. For example, the secret key for symmetric-key encryption, the random value y in the Schnorr scheme, and the private key in public-key primitives. The quantities that are generated must be of the required size and have sufficient entropy, i.e. the adversary should not be able to predict the upcoming values. To illustrate the importance of the required size, lets assume the adversary wants to brute force an AES encryption key. If a key of 128 bit is chosen, the adversary would on average have to test 2^{127} keys, since, there are in total 2^{128} possibilities. On the other hand, if a base secret of 2^{32} bits would be established, and, be expanded to an 128 bit key via a publicly known function, then the adversary can now guess the key on average in 2^{31} tries.

A (true) random number generator (TRNG) is defined by Menezes et al. [67] as "a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits" and a Pseudo-Random Number Generator (PRNG) as "a deterministic algorithm which, given a truly random binary sequence of length k, outputs a binary sequence of length $l \gg k$ which 'appears' to be random." Note that the input to the PRNG is referred to as the seed. While PRNGs are not as random as TRNGs, it can generate random quantities much faster. Therefore, the output of a TRNG is often used as the seed of a PRNG. A TRNG requires a naturally occurring source of randomness, e.g. electric or acoustic noise.

In NIST publication 800-90A [8], recommendations regarding the selection of PRNGs can be found. In this thesis, we will use the Hash-DRBG algorithm as a source for random numbers. Furthermore, we use SHA256 as internal hash function. For details regarding Hash-DRBG, we refer to the NIST publication.

2.3.5 Key derivation functions

The goal of a Key Derivation Function (KDF) is to derive one or more cryptographically strong secret keys from an initial source of keying material. This initial source of material can be e.g. a shared secret established via a DH exchange and additional entity identifiers. On its own, these input materials may not be distributed uniformly or an attacker may have insight on or control of specific parts. Therefore, the simple Hashed Message Authentication Code (HMAC)-based KDF (HKDF) [56], is used in this thesis to create strong secret keys. In practice, it is used in the Transport Layer Security (TLS) protocol version 1.3 [86]. Furthermore, it follows the extract-then-expand paradigm. First, the extract stage maps the input to a short pseudorandom key to concentrate the entropy of the input. Then, the expand stage maps the shorter key to the desired amount and length of pseudorandom keys. Internally, a hash function like SHA256 is used to compute the extract and expand stages.

2.4 Data communication

Secure data communication is required to protect the confidentiality, integrity, etc. of data. However, it is important to know how data is communicated between two entities. We start, therefore, with a high level overview of the required protocols to provide data communication between two entities. Then, an overview is given of the concepts used in this thesis to provide end-to-end security.

2.4.1 OSI model

A range of protocols are required to provide communication between two entities, as indicated by Figure 2.6. The figure represents the functions required to communicate using the Open Systems Interconnection (OSI) model. The OSI model is published by the International Organization for Standardisation (ISO) in ISO/IEC 7498-1:1994 [106], and, it provides an abstract reference model for communication networks. Each layer in this model serves the layer above it and is served by the layer below, and vice versa. A brief explanation of all layers is given in the following paragraph.

The **Physical** layer concerns the transmission and reception of data over a physical medium. The **Data link** layer provides the flow control and reliability of the communication over a physical medium. The two previous layers are typically defined in one standard, e.g. IEEE 802.11 (Wi-Fi). Next, the **Network** layer ensures the interconnection of different devices within a network. For example, the IP protocol is used to create the Internet as we know it with millions of connected devices. The route that a message takes is defined by these three layers, thus, the message is also processed by each entity in that route. The **Transport** layer provides the reliability of communication within a network, since, a message in transit may get lost or discarded. For example, the Transmission Control Protocol (TCP) ensures the reliability by, first establishing a connection, and secondly using acknowledgements. If a message is not acknowledged within a predefined time period by the remote entity, the message will be send again. The communication channel that is created using the previous layers is maintained by the **Session** layer. Using e.g. sockets to control the communication channel. The **Presentation** layer deals

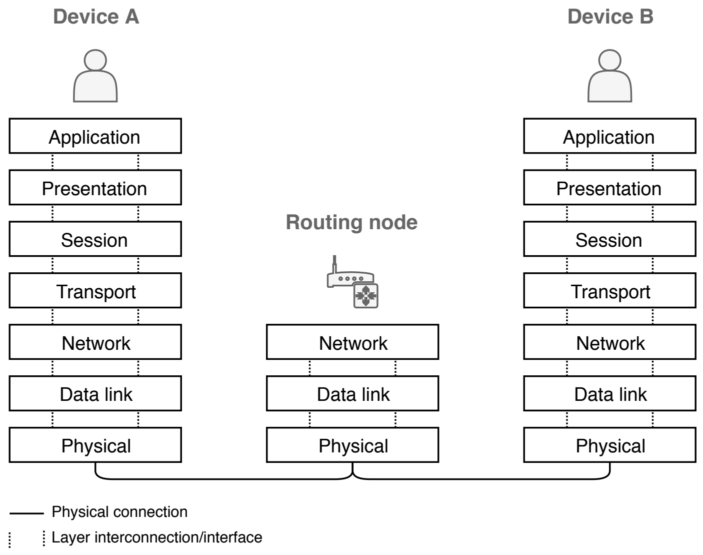


Figure 2.6: The interconnection between layers in the OSI model of a simple network.

with the representation of the information that is being transferred, i.e. the binary format of data. Finally, the **Application** layer details the services that are available and defines the required interactions between the communicating entities. For example, the Hypertext Transfer Protocol (HTTP) is used to retrieve websites.

2.4.2 Security

In almost all layers of the OSI model, security algorithms could be used to protect the communication channel. However, as mentioned earlier, the physical, data link, and network layer are typically processed by each entity that is used to deliver a message from one to another device. Thus, security protocols defined for one of these layers only provide hop-to-hop security, as shown in Figure 2.7. To illustrate, the IoT device communicates via the 6LoWPAN network and sends a message to a remote device on the internet, consequently, an intermediate node (IoT gateway) needs to translate 6LoWPAN messages to internet suitable messages (IPv4, ...). Thus, the security protocol would be processed at all three entities in this example. Therefore, confidentiality and etc. can not be ensured between the IoT device and the remote device.

End-to-end security provides the security guarantees between two communicating entities. It can be provided by protocols corresponding to the Transport layer and higher, since, they are only processed by the communicating endpoints (IoT device, Remote device). In this thesis, security at the transport and presentation layer are considered. Security at the transport layer protects the communication between two devices, for example via TLS. It secures all data that are generated by protocols belonging to a higher layer in the OSI model. Furthermore, TLS is the most reoccurring security protocol in practice. On the other hand, security at the application layer can only protect the piece of information (object) that is transmitted. This is denoted as Object Security.

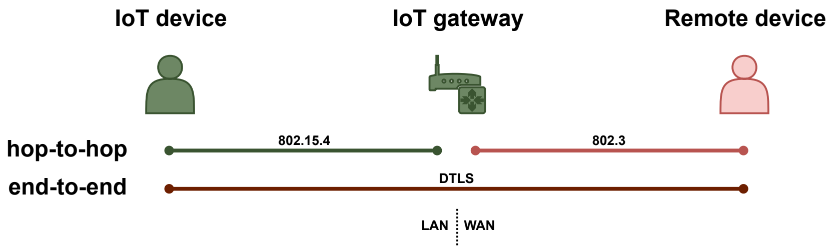


Figure 2.7: Example scenario of a hop-to-hop and a end-to-end secured communication channel.

2.5 Measuring energy consumption

Joule (J) is the derived unit of energy according to the International System of Units (SI) [111]. It is defined by SI as the energy required to move one object one meter using a force of one newton. Another definition is the work required to move 1 C through 1 V. However, the work required to produce 1 W over 1 s is a more suitable definition for an electrical environment.

In practice, the capacity of an electrical battery is typically expressed as Ampere-hour, see Figure 2.8. Also, one Ampere-hour is equal to 3600 C, as one coulomb is equal to one Ampere-second. The amount of energy in Joule (E) can be calculated by multiplying the the electrical potential difference (U) with the capacity defined in Ampere-hour (Ah), as shown in Equation (2.2). For example, the smallest battery in Figure 2.8 has a capacity of 750 mAh, thus, it has a capacity of 3240 J.

$$E = 3600 * Ah * U \tag{2.2}$$

Figure 2.8: Example of an AA and AAA battery with a supply voltage of 1.2 V. The AA and AAA battery have respectively a capacity of about 750 mAh and 1900 mAh.

Generally two methods are used to measure the energy required by an electronic board: via continuous physical measurements or performance measurements. These two methods will be described in more detail hereafter.

2.5.1 Via physical measurements

The most accurate method for measuring the required energy of an electronic board or chip is via the continuous measurement of the electrical current that is drawn by that board. This is typically done using a shunt resistor. This is a low-resistance resistor that is either placed at the source or drain of the energy source, see Figure 2.9.

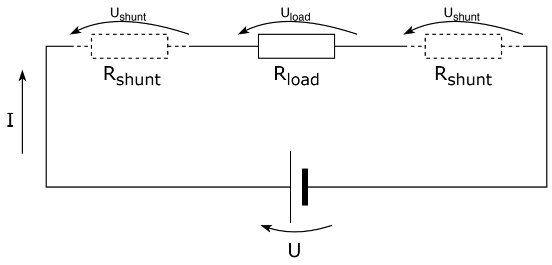


Figure 2.9: Schematic representation of the placement of a shunt resistor.

The total voltage that is supplied by the energy source is divided over the shunt and the load resistor. To not influence the load, the resistance of the shunt resistor should be much smaller than the load resistance. The voltage (U) needs to be measured over the shunt resistor via e.g. an Analog-to-Digital Converter (ADC). Then, the current (I) can be calculated using Ohm's law, since, the resistance value (R) of the shunt is known. Using the current and voltage value, the power (P) drawn by the load can be calculated using Equation (2.3). Finally, the energy consumed (E) by the load is the integral of the power over a specific time period, see Equation (2.4).

$$P(t) = U * I(t) \tag{2.3}$$

$$E = \int_{t1}^{t2} P(t)dt \tag{2.4}$$

2.5.2 Via performance measurements

The energy consumption of an electronic board or chip can also be estimated via the reported electrical characteristics, which can be found in the chip's respective data sheet. Consequently, it is less accurate than the physical energy measurement, but, it is typically easier to implement and use. Note that a data sheet is a report made by the manufacturing company that describes the electronic board or chip. It contains electrical characteristics like recommended supply voltage, maximum output current and average power consumption (P_{avg}). To estimate the energy consumption, first, performance measurements are done to identify e.g. the time a chip spends in active mode. An example time characteristics of an electronic chip is presented in Figure 2.10. Then, the average power consumption that corresponds to that action is multiplied by the measured time period (T) to calculate the energy consumption (E), see Equation (2.5).

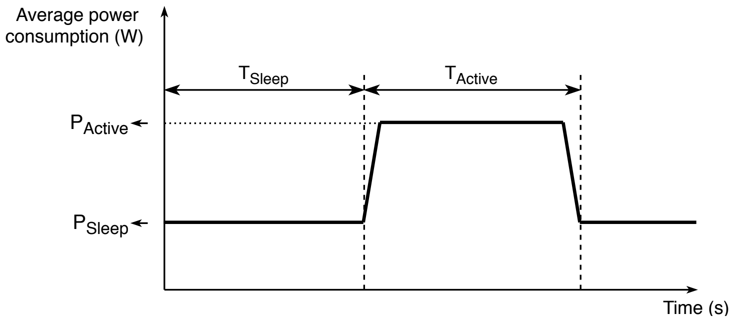


Figure 2.10: Example time characteristics of an electronic chip.

$$E = P_{avg} * T \tag{2.5}$$

Chapter 3

End-to-end secured communication for heterogeneous IoT networks via fog computing

This chapter describes the HeComm (Heterogeneous Communication) architecture [126]. The HeComm architecture utilises fog computing and object security to provide end-to-end secured communication between IoT nodes residing in networks that operate over different communication protocols. The chapter is organised as follows. After an introduction in Section 3.1, the required background knowledge on fog computing is provided in Section 3.2. In Section 3.3, related work on the interconnection of heterogeneous IoT nodes and on object security is described. This is followed by a description of the HeComm architecture in Section 3.4. In Section 3.5, we give a high-level description of the communication flow in our proof-of-concept implementation. Section 3.6 provides more details on this implementation. The results are evaluated in Section 3.7. Finally, the chapter is concluded in Section 3.8.

3.1 Introduction

The IoT introduces the challenge of connecting devices to each other and to the cloud while also providing cryptographic protection to the vast amount of data that are generated. The heterogeneity of the IoT poses an additional challenge, since a lot of IoT networks utilise different communication protocols and physical interfaces, depending on the features required by the application. For example, networks based on LoRaWAN [104] can be used to provide long-range communication, whereas 6LoWPAN [69] is suitable for IP-based communication in personal area networks. These two types of networks are not compatible because of the difference in physical interface, communication protocols, etc. Furthermore, utilising the same protocols does not necessarily mean enabling interconnection. For example, two 6LoWPAN networks hosted by different entities cannot communicate with each other, because, the encryption keys used in the link layer are managed separately.

The interconnection of heterogeneous networks has been done before through various methods, e.g. IoT gateways, but are often lacking end-to-end security, i.e. the gateway typically decrypts and re-encrypts the data packets when they are transferred from one network to the other. With the HeComm architecture, we provide a method to enable the interconnection of heterogeneous IoT nodes while also providing an end-to-end security guarantee. Our solution uses object security to secure the communication flow. This is introduced at the level of the application layer, which is often not specified by the network's communication standard.

My contribution to the work presented in this chapter was the design and implementation of the architecture as well as the practical evaluation.

3.2 Background on fog computing

The fog computing paradigm is rising in popularity but has not yet been globally standardised. An effort is underway from the OpenFog Consortium [76] that has the backing of major companies, e.g. CISCO, and universities. However, a reference architecture is not yet available. We provide, therefore, our own interpretation and usage of the fog.

The fog topology can be divided into the following two parts: the fog entities and the fog server. The fog entities are located at the edge of the local network and are in direct contact with the IoT networks, as shown in Figure 3.1. Thus, the fog entities provide services to the IoT, e.g. data filtering. Overall, the fog entities can communicate with the IoT networks, with each other, and with the cloud. These devices are less powerful than cloud servers and less constrained than IoT devices, e.g. routers. The fog server is a cloud server, manages the fog entities, and acts as a certificate authority (CA). Each fog entity is provided with a certificate signed by the fog server. Consequently, the identity of each entity can be verified by the devices using the services of the fog. The challenge that we tackle in this chapter is the end-to-end secured communication of heterogeneous IoT services. The HeComm solution that we propose utilises fog entities for this purpose.

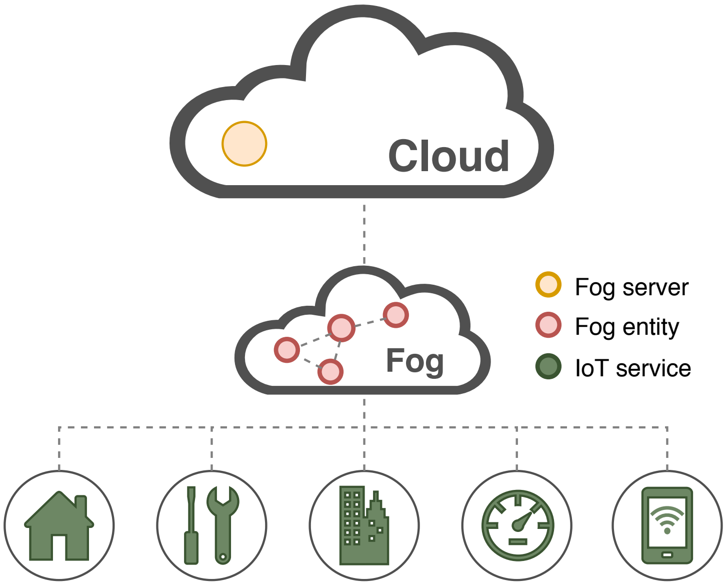


Figure 3.1: Representation of a fog network.

3.3 Related work

The work of Cirani et al. [21] proposes the use of a fog node as an IoT hub that provides the abstraction and interconnection of the Constrained Application Protocol (CoAP) services of the heterogeneous constrained devices. This IoT hub can act as a gateway between IoT networks, provide service discovery, etc. The idea of using the fog nodes to provide interconnection is similar to that of ours and can provide many functionalities to assist in the IoT-IoT or IoT-cloud communication. In addition, our HeComm architecture concentrates on guaranteeing the security of the communication. It relies on the fog to facilitate heterogeneous communication, but reduces the trust in the fog to obtain end-to-end security between IoT nodes.

The architecture of Van Den Abeele et al. [115] does not utilise the fog but proposes to access the power of the cloud to provide an IoT abstraction layer. This enables easier connectivity, since the cloud is used to provide the abstraction. The cloud, however, does not allow low-latency communication, while the fog can provide it for IoT-IoT communication. Furthermore, by abstracting the services through the cloud, the origin of the data cannot be authenticated. In our architecture, we provide end-to-end secured communication between the supplier and the user of the IoT service.

With regards to the use of object security in constrained devices, two publications are worth mentioning. In the master's thesis of Brorsson et al. [15], a comparison was made between Datagram Transport Layer Security (DTLS) and Object Security of CoAP (OSCOAP). The result was that OSCOAP is comparable to DTLS in terms of performance, memory footprint, etc. Another example is the work written by Navas et al. [72] that uses and extends the CBOR Object Signing and Encryption Protocol (COSE) protocol. They propose an authenticated key establishment protocol over OAuth 2.0 for the IoT. Our work further analyses the use of object security in a security architecture for constrained nodes, with a specific focus on heterogeneous networks.

3.4 HeCommm architecture

The HeComm architecture can be divided into two parts: the HeComm protocol and the end-to-end secured communication. The HeComm protocol establishes the bridge between the two networks and the secret key that will be used by the IoT nodes for the end-to-end secured communication.

The general topology of the HeComm architecture comprises of a number of IoT networks that operate over different communication protocols, each consisting of a number of IoT nodes. A fog entity (fog node) takes care of the connection between the networks. This is presented in Fig. 3.2, which, for simplicity, only depicts two different IoT networks, with one IoT node in each network. By using the proximity of the fog node to the IoT networks, the fog node can, if the IoT network allows it, directly communicate with the nodes. The identity of a fog entity can be verified via its certificate, which is provided by the fog server.

The setup of the secure communication, i.e. the HeComm protocol, is executed by the network managers of the IoT networks (NwkServer in Fig. 3.2) and the

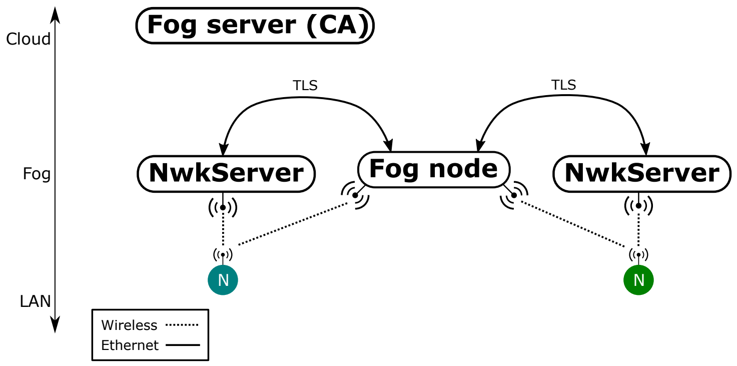


Figure 3.2: Simplified representation of the general network topology of the HeComm architecture.

fog node. The network managers manage their respective IoT nodes and are therefore also capable of providing cryptographic material for the managed IoT nodes. An example can be found in the LoRaWAN specification, where a key derivation scheme is used between the application server and the LoRa node to create the AppSKey and NwkSKey. We refer to Chapter 2 for more background information on LoRaWAN. Furthermore, IoT nodes have limited resources and it is therefore beneficial to outsource the key establishment to devices with less constraints that can utilise stronger algorithms like asymmetric cryptography. During this establishment, the fog node will act on and relay parts of the HeComm protocol because of its centrality in the entire communication scheme. As a consequence, the HeComm protocol relies on the availability of the fog node, but the fog node will not be able to decrypt the exchanged messages.

3.4.1 End-to-end secured communication

Two entities can only communicate if they use the same communication protocol. In a heterogeneous environment, this is almost never the case. Therefore, we use the fog as an IoT gateway that can understand both communicating parties. The IoT gateway provides a bridge and will enable communication between the two IoT networks by translating mismatching protocols.

Finding a common ground between two heterogeneous protocols facilitates the communication. An example is given in Fig. 3.3, in which the communication stack of 6LoWPAN and LoRaWAN are mapped onto the OSI model. Notice that, for both IoT networks, the specified communication protocols only reach until the Network layer. For 6LoWPAN, the highest layer specified is the Network layer, where the 6LoWPAN header compression is used. In the LoRaWAN network, the specifications can be matched until the Transport layer. The LoRaWAN specification provides a UDP-like functionality with the FPort field. The other layers, Session till Application layer, can be chosen by the developer.

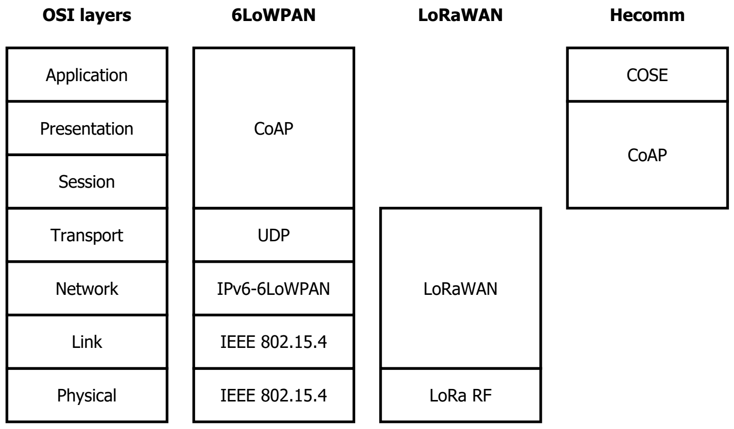


Figure 3.3: 6LoWPAN and LoRaWAN networks mapped onto the OSI model.

The common ground between these two IoT networks is the Session till Application layer. An often used protocol for these layers, in constrained networks, is the Constrained Application Protocol (CoAP) [100]. CoAP is an application layer protocol that is designed as the counterpart of the HTTP protocol but for constrained devices. The recommended security protocol for CoAP is DTLS. However, DTLS is created with the intent of securing a UDP connection, which does not match with our goal to use protocols belonging to the Session till Application layer. Another method to secure CoAP is the Object Security for Constrained RESTful Environments (OSCORE) standard proposed by the IETF [97]. Initially it was called OSCOAP, but, it has evolved to OSCORE. However, a stable implementation is not yet available. In our work, the CBOR Object Signing and Encryption Protocol (COSE) [95] protocol concept of the OSCORE standard is used in this architecture to provide end-to-end object security. COSE is a proposed standard of the IETF, and, a stable implementation is available (cose-c [25]). It can provide signatures, cryptographic keys, message authentication and encryption using CBOR [12] for (de)-serialisation. The IoT nodes in this architecture use CBOR for the (de)-serialisation of the application data. It is designed to have a small code size and small message size, suitable for IoT networks. For securing this data format, COSE in combination with AES-CCM AEAD block cipher mode is used to provide authentication and confidentiality. This is achieved by using COSE objects as CoAP payload.

3.4.2 HeComm protocol

The HeComm protocol can be described as establishing a link contract between two IoT nodes in a heterogeneous network. Fig. 3.4 shows the protocol for the establishment of a link contract between two generalised IoT networks. It provides two functionalities: service agreement and key agreement. The service agreement ensures that all parties agree upon a link between two IoT nodes, i.e. the nodes that will be communicating with each other. Secondly, a secret key is generated via an Authenticated Key Establishment (AKE) protocol. The resulting secret key will be used to secure the communication between the IoT nodes.

The IoT networks have one network manager (NwkServer) and one IoT node (Node A or Node B) in the generalised setting. The link contract will be used by the fog entity to relay the communication of the respective IoT nodes. Only the network managers of the IoT networks and the fog node will participate in the HeComm protocol. Furthermore, these entities must have and use a certificate signed by a trusted third party, e.g. the fog server's CA. This will be used in the secured TLS connection between the network managers and the fog to verify each other. The communication between the network managers and their respective IoT nodes are not defined by the HeComm protocol since they depend on the communication protocol used by the IoT network. Note that the networks managers only communicate with the fog node and not directly with each other. It is, therefore, easier for the network managers to implement and manage the HeComm architecture.

The communication during the HeComm protocol is JSON-based. The schema of the top-level JSON object is comprised of two fields: the FPort and Data field. The FPort field is used to indicate one of the following messages: *DBCommand*, *Link Request*, *Link State*, *Link Set* or *Response message*. The *DBCommand* message is used to provide configuration material to the fog node to enable communication between the fog node and the corresponding IoT network. This type of message is not used during the HeComm protocol. In the messages *Link Request* and *Link Set*, a Link Contract in the form of a JSON object is encoded into the Data field. The Contract contains four fields: InfType, ReqDevEUI, ProvDevEUI and the Linked field. The InfType field of the Link Contract object describes the type of information requested, but since this is a proof-of-concept implementation, this field was not extensively defined and is indicated by a number. This field should indicate the type of information ranging from a more general data type, e.g. temperature, to the specification of a specific IoT network and IoT node. The ReqDevEUI and ProvDevEUI field contain, at the end of the protocol, the unique identifiers of the respective IoT node. The last field, Linked, is used to indicate if both network managers have successfully established a secret key. For the message of the type *Link State*, the data field will contain the messages of the key agreement protocol. To acknowledge or deny a request to form a Link Contract, the *Response message* is used. In the *Response message*, the data field contains a Boolean value corresponding to the intended response. All the general message types, except for the *DBCommand*, are used during the protocol. The functionality of these messages will be clear after the description of the data flow during the HeComm protocol.

The *DBCommand* is used by the network managers of the IoT networks to notify the fog and configure it for the HeComm communication. It can be used to provide configuration material for the fog to allow communication with the IoT network. Furthermore, it must be used before the HeComm protocol to provide the fog node with the details of the IoT nodes that will be using the HeComm communication, e.g. providing the device's unique identifier. This enables the fog to communicate with the IoT network and correctly relay the messages.

The initiator of the HeComm protocol, shown in Fig. 3.4, is the IoT network manager of node A requesting some type of information, as of now called the requester. This requester will send a *Link Request* message (LC(A,X)_req) containing, at minimum, the InfType and ReqDevEUI field defined. The fog node will act on this request by searching for a suitable provider node. In this setting, the network manager of node B is the provider. The fog node will fill in the provider node into the Link Contract ProvDevEUI field and relay the Request message with the Link Contract to the provider (LC(A,X,B)_req). The provider can then choose to accept or deny the Link contract, via a *Response message* (Ack_resp). Now, the requester and provider use an AKE to agree upon a master secret. During the execution of this protocol, only *Link State* messages are sent and the fog will only relay these messages without altering them. The established master secret will be used as the secret key (K_{OS}) to secure the communication between the IoT nodes. In the last phase of the HeComm protocol, it is required for the requester to request the link to be set in the fog node, using the *Link Set* message with a completely filled in Link Contract (LC(A,X,B,Ok)_set). The fog node will then verify this Link Contract with the provider and enable the link between the respective IoT nodes. Now both parties can push the agreed upon secret key to their IoT node to enable the communication (set(K_{OS})).

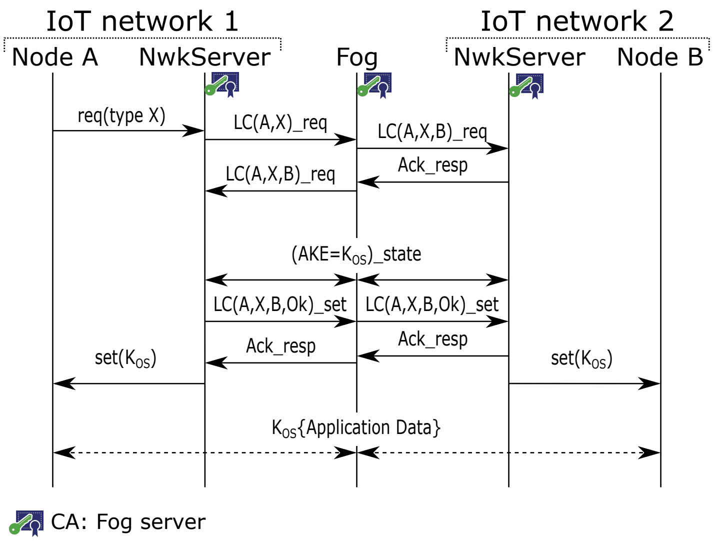


Figure 3.4: The data flow of the HeComm protocol and in IoT network messages. All participants of the HeComm protocol (NwkServers and fog node) have a certificate signed by a trusted party, the fog's CA in this case.

3.5 Proof-of-concept system

The HeComm architecture is validated using a proof-of-concept implementation using the 6LoWPAN and the LoRaWAN network. In this section, we discuss the communication interfaces that are required between the fog and the IoT networks, namely LoRaWAN and 6LoWPAN. Similar specifications can be provided for other types of networks. The most important requirement is that the heterogeneity of the network should not have an impact on the communication between the IoT nodes.

The fog nodes are devices that are located close to the IoT networks. To allow IoT-fog communication, some specifications need to be set, depending on the type of IoT network. We assume that the fog has an Ethernet interface for communication with the cloud and network managers, but also an interface to communicate directly with the IoT network, e.g. an IEEE 802.15.4 interface.

3.5.1 LoRaWAN

A basic topology of the LoRaWAN network consist of an Application server, Network server, gateway (LoRa_gw), and the IoT nodes (LoRaWAN nodes). The application server is the owner of the node and the network manager in our HeComm protocol. The Network server is the backbone of the LoRaWAN network and provides network related functionalities like routing. The gateway provides the translation from wireless to wired communication. All the previous entities are used by the LoRaWAN node to communicate with the Application server to e.g. upload sensor data. We refer to Chapter 2 for more details on the LoRaWAN network.

Communication between LoRaWAN based IoT nodes is not allowed. The IoT nodes only communicate with the Network and Application server. To ensure compatibility with existing IoT networks, the specification of the network should not be altered to the benefit of the HeComm architecture. Therefore, direct communication between the LoRaWAN node and the fog node is not possible. Instead, the LoRaWAN interface leverages the centrality of the Network server in the LoRaWAN network. By utilising the Network server, the communication also benefits from the integrity protection the NwkSKey provides. In this setup, the proposed future standardised application extension of the FPort field in the Frame Header is used to enable the Network server to relay packets to the fog node instead of the Application server. This way, the fog node will act as an Application server in the LoRaWAN topology. The FPort field with the value of 255 indicates communication coming from and to the fog node (HeComm communication). Furthermore, the FPort value 254 is also used. The Application server uses it to push the secret key of the HeComm communication to the IoT node.

3.5.2 6LoWPAN

A 6LoWPAN network typically consist of a gateway (6LoWPAN_gw) and the IoT nodes (6LoWPAN nodes) in a mesh network configuration. The gateway provides all network management functionalities and acts as the network manager in the HeComm protocol. For more information regarding 6LoWPAN, we refer to Chapter 2.

The 6LoWPAN network does allow communication between IoT nodes. We therefore have provisioned the fog node with a physical interface that can relay 6LoWPAN packets between the IoT network and the fog node. This interface will act as a regular IoT node in the 6LoWPAN network. It will receive its own IP address from the RPL root and can relay messages in the mesh network, if needed. For all the nodes using the HeComm service, the interface looks like a regular node within the network. The nodes are, therefore, not aware of the heterogeneous communication. The packets destined for the interface will be relayed into the fog node and the fog can use this interface to send packets to other 6LoWPAN nodes. The downside of the direct communication in the IoT network is the need of the Network key. The Network key is used for the security of the IEEE 802.15.4 link layer. This key needs to be provisioned to the fog node using the DBCommand message. In our implementation, this key is static and is hard-coded in all the devices.

3.6 Implementation

The network topology of our proof-of-concept implementation is presented in Fig. 3.5. On the left side, a simple LoRaWAN network topology is provided and on the right side a simple network topology of a 6LoWPAN network. In the proof-of-concept, the TLS handshake using the ECDHE_RSA cipher suite is used as AKE protocol. This ciphersuite uses the ephemeral version of Elliptic-curve Diffie–Hellman (ECDH) with RSA-based certificates (X.509). The X.509 certificates are used by the requester and provider to verify each other's identity. Note that other AKE protocols or ciphersuites could be used. The pre-master secret resulting from the TLS handshake is pushed to the IoT nodes as secret key.

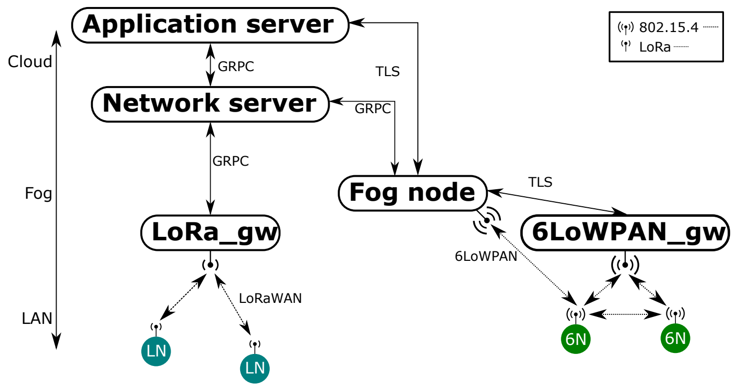


Figure 3.5: Network topology of the proof-of-concept implementation.

The fog node is implemented on a Commercial off-the-shelf (COTS) router, Linksys WRT1200AC. This router runs the OpenWRT OS to enable third party applications. Our fog node application, shown in Fig. 3.6, implements multiple interfaces and is cross compiled for ARM. To facilitate the implementation, the program is written using the Go programming language. The fog core is the main thread which implements the HeComm protocol and uses a MySQL database and the LoRaWAN and 6LoWPAN interface to relay the packets to the right IoT nodes. It uses a TLS interface on the Ethernet connection to listen for HeComm messages. The LoRaWAN interface is also concurrently listening on the Ethernet connection for LoRaWAN packets coming from the LoRaWAN Network server. The physical interface to connect with the 6LoWPAN network is a Zolertia Z1 node [134], running Contiki [24], connected to the COTS router via USB using the Serial Line Internet Protocol (SLIP). The Zolertia Z1 node implements the 6LoWPAN and IEEE 802.15.4 protocol and communicates with the COTS router using non-compressed IPv6 packets over the SLIP connection.

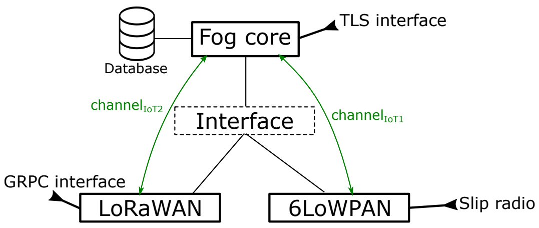


Figure 3.6: Diagram of the implementation of the fog node.

The 6LoWPAN network contains one IoT node and a network manager. The network manager is a Dell Precision M4800 laptop with a USB-connected Zolertia Z1. The connected Zolertia node runs Contiki and is based on the rpl-border-router implementation, found in the online Contiki examples. This node acts as a border router and provides IP addresses to the nodes in the network. The laptop implements the HeComm protocol and uses the Zolertia node to communicate with the IoT node in the 6LoWPAN network. This IoT node is also a Zolertia Z1 node running Contiki. It implements the IoT node functionality, requesting data from the network manager and sending CoAP requests via the fog node. It uses the cose-c [25], cn-cbor [16] and er-coap [55] library to implement the HeComm protocols. The AES-CCM cipher mode used in the COSE protocol has been mapped onto the AES-CCM hardware accelerator which is also used for link layer security. The Zolertia Z1 node has 92 kB of ROM and 8 kB of RAM.

The Application server, Network server and the gateway of the LoRaWAN network are implemented on one Raspberry PI 3 MODEL B. This Raspberry PI runs on Raspbian and uses the IMST iC880A LoRaWAN concentrator board as physical interface to the LoRa network. The three applications running on the Raspberry PI communicate with each other via the loopback interface. The loopback interface is a virtual interface that enables IP communication within the device. The three applications are developed by the open source LoRa server components (https://www.loraserver.io/). Only the Network server and the Application server have been modified by us to implement the required communication for HeComm. The IoT node in the LoRaWAN network uses the mbed-os on a STM32 NUCLEO-L073RZ node with a LoRa Shield as physical interface. It only acts as a provider HeComm node and will answer the CoAP request originating from the 6LoWPAN node. This node uses the publicly available cose-c, cn-cbor, mbed-coap and Mbed TLS libraries to implement the HeComm communication and security protocols. The Mbed TLS library is used to provide the AES-CCM algorithm and cipher mode. The STM32 NUCLEO node has 192 kB of flash memory and 20 kB of RAM.

3.7 Evaluation

The results discussed in this section cover both the security evaluation and the implementation results of the IoT nodes. We only focus on the IoT devices, since the other devices in the proof-of-concept implementation are not necessarily constrained. Next, only the impact on memory (RAM) and storage (ROM) are considered. In a typical scenario where two nodes are communicating within the network, the nodes will receive and use their encryption key to encrypt or decrypt the communicated messages. In our HeComm architecture, the encryption key is also pushed to the device. However, the messages are not send directly to the respective recipient, as the fog is used to provide the interconnection of the two IoT nodes. In both scenarios, the nodes receive and use the encryption key to secure the communicated data using similar algorithms. Thus, adding the HeComm architecture will primarily require additional firmware and memory on the IoT nodes to implemented the required communication protocols. Though, an in-depth study of the energy impact of the AES encryption algorithm in terms of communication and computation is done in Chapter 5.

3.7.1 Security evaluation

The security of the communication is evaluated using an informal analysis. First, the threat model and assumptions are given. Then, the HeComm protocol is assessed. Finally, the data flow of the HeComm communication, which deals with the communication between the IoT nodes, is examined.

In general, we consider two network managers, two IoT nodes, and the fog node. In our proof-of-concept, we have to take into account the following seven entities: LoRaWAN application server (network manager), LoRaWAN network server, LoRa gateway, LoRaWAN node, fog node, 6LoWPAN gateway (network manager), and 6LoWPAN node. Note that there may be multiple LoRaWAN and 6LoWPAN nodes. The adversary is assumed to know the protocol and may eavesdrop, modify, corrupt, delete, redirect, or replay all the messages that are transmitted. Additionally, the fog node may act maliciously and try to illegitimately obtain the data that is communicated. Furthermore, we assume that the specifications of the IoT networks provide secured communication within their network. The goal of the adversary can be to illegitimately obtain the data that are communicated or to degrade the communication service.

All entities participate in the **HeComm protocol**. First the network managers of the IoT networks and the fog node agree upon a link contract. This link contract is ratified by the network managers through establishing a shared key via the TLS protocol, where the fog node acts as a relay node. In this first phase of the protocol, an attacker is unable to eavesdrop, modify, etc. the link contract, because, the network managers and the fog node communicate over a mutually authenticated TLS secured communication channel. The network managers and the fog node can verify the identity of each other via the available certificates. Entities that have become malicious should be revoked by the certificate authority, which is the fog server in this proof-of-concept. If an adversary is the malicious fog node that is participating in the protocol, it can degrade the communication service by e.g. not relaying packets. In this case, the network managers can report the malicious fog node, consequently, the fog server can revoke this specific fog node. Next, the malicious fog node cannot intercept the key established between the two network managers, as the TLS protocol is secured against man-in-the-middle (MITM) attacks and provides mutual authentication. Also, the chosen TLS ciphersuite uses the ephemeral version of the ECDH protocol which provides perfect forward secrecy. This ensures that a compromised key cannot compromise previously established keys. In the final phase of the protocol, the network managers push the established key to the respective IoT nodes via their respective relay devices. This transaction is assumed to be secured via the security architecture of the IoT networks.

The COSE protocol is used to secure the **HeComm communication**. By using the AES-CCM cipher mode, data authentication and confidentiality is guaranteed for the payload communicated between the two IoT nodes. Thus, the attacker cannot, without knowing the secret key, understand or alter the payload. This is also the case for the fog node, which cannot alter or change the payload of the messages. The attacks prevented are therefore eavesdropping attacks and man-in-the-middle (MITM) attacks. However, using only COSE in the payload does not guarantee protection against e.g. replay attacks, which OSCOAP would protect against. One countermeasure is provided via the AES-CCM cipher mode, since a message with a reused IV should be dropped. Additionally, replay protection is often guaranteed via the security architecture of the IoT networks. An attack a malicious fog node could be more successful at than an outside attacker is a Denial-of-Service attack. The message originating from the fog node is always processed, since it bypasses the network security. This, however, should be easily detectable and the fog node should in that case be revoked by the fog server. In the proof-of-concept the LoRaWAN and 6LoWPAN network are used as communicating networks. The network security is provided by the IEEE 802.15.4 link layer security in the 6LoWPAN network and the integrity protection of the NwkSKey in the LoRaWAN network. Moreover, the potential impact of the LoRa gateway in this data flow is limited, as it only provides the relay service between the wireless and the wired network. In summary, the fog node is not able to illegitimately obtain the data that is communicated. However, it could degrade the communication service, but not without the risk of being detected.

3.7.2 Implementation overhead

In Table 3.1, the resource usage of both IoT nodes are presented in terms of RAM and ROM usage (bytes). The results for 6LoWPAN are from the analysis of the binary compiled for the Zolertia Z1 node, using the MSP430-size command. The resource usage of the LoRaWAN node comes from the output results after compilation using the mbed Command Line Interface (mbed-cli). In the table, the distinction has been made between resources used for the Object security (COSE + CBOR + AES-CCM), for the CoAP library and for the HeComm-specific code. These values are the result of the difference between the implementations, with and without the respective library. Lastly, the bare implementation, without HeComm communication, and the full-implementation resource usage are presented. The RAM usage of the separate libraries of the mbed-os for the LoRaWAN node are particularly low. We expect the mbed-cli does not take into account the RAM occupied by the libraries. Nevertheless, since the implementation of 6LoWPAN uses the same objects from similar libraries as the LoRaWAN implementation, we can expect around the same RAM usage for the libraries in both implementations.

The full implementations of both networks report under 90 kB of ROM and 7 kB of RAM usage. We compare our results to the classification of constrained nodes published by the IETF [11]. Class 0 nodes have lower than 100 kB of ROM and 10 kB of RAM and Class 1 devices have around 100 kB of ROM and 10 kB of RAM, according to the IETF classification. Therefore, the results can be mapped to Class 1 devices, which are described as devices which cannot easily directly communicate with the internet but can communicate

Table 3.1: Implementation results, RAM and ROM usage, of the IoT nodes.

Implementation	**6LoWPAN**		**LoRaWAN**	
	ROM (B)	RAM (B)	ROM (B)	RAM (B)
Object security	8310	582	22836	56
CoAP	12010	622	7048	204
Hecomm	1654	204		
Bare	45727	5176	57321	5168
Full	67701	6584	87205	5428

using protocols specifically designed for constrained nodes. In the case of the 6LoWPAN node, the AES-CCM cipher mode is already supported in the standard hardware/software implementation. In other IoT networks, the AES algorithm is often used in CBC, CTR or even CCM mode. Re-using the already available AES-CCM cipher mode leads to a smaller implementation of the object security for the 6LoWPAN node than for the LoRaWAN node. The difference is 14.5 kB of ROM usage.

3.8 Conclusion

In this chapter, we describe the HeComm architecture that provides secured end-to-end communication between IoT nodes in heterogeneous networks. The proof-of-concept implementation with the interconnection of the 6LoWPAN and LoRaWAN network proves the feasibility of our solution. The overhead caused by the HeComm-specified protocols is minimal, since a combination of popular protocols and algorithms are used, that are usually already foreseen in secured constrained devices. The IoT nodes supporting the HeComm architecture can still be classified as Class 1 constrained nodes. The security evaluation of the HeComm architecture shows that the COSE protocol can guarantee end-to-end data security. The fog is used to provide the service of heterogeneous interconnection, but the fog entities are not able to compromise the security of the communication between the IoT nodes.

Chapter 4

Trade-offs between storage and computation for embedded digital signatures and signcryption schemes

This chapter evaluates the coupon technique in an IoT setting, and, it is based on [120, 123]. Four optimisations and memory encryption are considered to analyse the trade-off between storage and computation. The coupon technique is applied to two schemes: Schnorr [96] and signcryption [14]. The chapter is organised as follows. Section 4.1 introduces the topic and highlights the contributions. Section 4.2 gives an overview of related work. More details on the applied storage and computation techniques and their implementation are presented in Section 4.3 and Section 4.4, respectively. In Section 4.5, the results are discussed, and Section 4.6 concludes the chapter.

4.1 Introduction

In this chapter, we apply the coupon technique, see Section 4.2 for more details, to optimise the storage and computation requirements of the following two cryptographic primitives: digital signatures on the one hand, and signcryption schemes, which are a combination of digital signatures and encryption schemes, on the other hand. More specifically, we concentrate on lightweight cryptographic schemes based on elliptic curve cryptography (ECC). Digital signatures are implemented using the EC-based Schnorr algorithm [96]. For the signcryption scheme, encryption is added to the Schnorr signature scheme [14]. Both schemes are described in more detail in Section 2.3.3.

Our contributions are as follows:

- We apply four techniques that reduce the data storage requirements for EC-based Schnorr and signcryption using the coupon technique: coupon encoding, point compression, index compression and coupon addition.
- We compare these four storage reduction techniques to an implementation that does not rely on the coupon technique but uses the comb method to reduce the online computation requirements for EC-based Schnorr and signcryption.
- We apply memory encryption to protect the stored (precomputed) values.
- We evaluate and compare the storage, performance, memory, and energy requirements of all these techniques on a health sensor platform.

My contribution to the work presented in this chapter was the implementation and evaluation of the different storage and computation techniques. The idea to perform this evaluation was initiated and fine-tuned in collaboration with my co-authors.

4.2 Related work

Our goal is to minimise the storage and the computation requirements of digital signature and signcryption schemes on IoT devices. This section elaborates on related work that also tries to achieve this goal.

In order to decrease the computation cost of cryptographic operations on constrained devices, a common approach is to offload the compute-intensive operations to more powerful nodes. This technique is applied by Saied et al. in [93] to the Transport Layer Security (TLS) handshake and the Internet Key Exchange (IKE) protocol, and in [92] to the Host Indentity Protocol - Base Exchange (HIP-BEX) scheme.

Another approach is based on the precomputation of a number of intermediate values in the cryptographic schemes that run on the IoT device; these values are then e.g. stored on the device before deployment. This technique is also referred to as the **coupon technique**. Lee et al. [60] apply this to the Schnorr signature scheme on a system consisting of a smart card and a terminal. This technique was further evaluated for both the EC-based Schnorr and signcryption scheme on an embedded device in [123] and [120].

In addition to the coupon technique, numerous methods have and are being devised to reduce the computation time of ECC operations, as described by Hankerson et al. [41]. For example, the **comb method** enhances the point multiplication operation by precomputing a window containing EC points related to a fixed base point. An modified version of this method was designed by e.g. Hedabou et al. [42] and applied in the Mbed TLS library. The work of Liu et al. [63] is another example that uses EC optimisation techniques to implement an ECC cryptographic library.

Our work implements and evaluates four storage reduction techniques for the coupon technique. Further, we compare the results of these techniques to an implementation that uses the comb method. Additionally, the computational overhead of memory encryption is evaluated.

4.3 Storage and computation methods

In total, six techniques are evaluated: four for storage optimisation when the precomputation of coupons is used (coupon encoding, point compression, index compression and coupon addition), one for computation optimisation when no precomputation is used (comb method), and one for protecting precomputed coupons (memory encryption).

4.3.1 Coupon encoding

An efficient way of representing EC points is SEC1 encoding [17]. We propose an extended format of the SEC1 encoding, as shown in Table 4.1. A Schnorr coupon consists of the B0 header, the random value, and the EC point coordinates (X1 and Y1). A signcryption coupon additionally requires the B1 header and the second EC point coordinates (X2 and Y2). The B1 header uses the same fields as the header of a SEC1 encoded EC point. The compressed and uncompressed field indicate EC point compression. If EC point compression is used, the parity of the Y coordinate is provided in the Parity field in the respective header and the Y coordinate field (Y1 or Y2) is removed. This compression technique is explained in Section 4.3.3. In contrast to the B1 header, three fields are added to the B0 header: Type, Rnd, and Rnd compression. The Type field identifies the scheme. The Rnd field is used to indicate the presence of the random value.

The Rnd compressed field indicates if the random value is compressed. This technique is explained in Section 4.3.4. The coupon is encoded as follows. First, the header $B0$ is added, which is then followed by, respectively, the first EC point coordinates and a random value. Finally, only in the signcryption scenario, the EC point coordinates that represent the encryption key are encoded using the SEC1 encoding with corresponding header $B1$ and are then padded to the random value.

Coupon encoding is always applied when storing a precomputed coupon, while the other techniques, explained in the following paragraphs, are evaluated separately or in combinations.

4.3.2 Memory encryption (e)

In the IoT device, the encoded coupons are stored in e.g. flash memory. They are stored in our implementation as a C/C++ array of coupons, where the index of each coupon is based on its relative position to the first coupon in the array. In summary, the system only keeps track of the pointer to the first coupon and the total amount of coupons.

Memory encryption ensures the secure storage of the coupons on the device. It will not utilise AES in ECB block cipher mode, as proposed by Winderickx et al. [123], but AES in CTR mode. Unlike ECB mode, which would require padding with the extended SEC1 format, data alignment is not an issue in CTR mode. Therefore, CTR mode is more efficient for storing the coupons. However, CTR mode requires an initialisation vector (IV) of 16 bytes. For our implementation, the IV is defined as follows: 12-byte coupon counter || 4-byte block counter. The coupon counter is initialised with a nonce of 12 bytes while the block counter starts from zero. The IV of a specific coupon in the list of stored coupons can be calculated by adding the index of the coupon to the nonce. The block counter is incremented every time a new IV is generated. The AES cipher in CTR mode will utilise this IV to generate the different cipher text blocks to perform the decryption. Note that the decryption key should be stored safely in the MCU, so that it is only accessible by this decryption process.

4.3.3 Point compression (c)

Instead of storing and transmitting both the X and Y coordinates of a point, the Y coordinate can be derived from the X coordinate, the definition of the

Table 4.1: Format of Schnorr and signcryption coupons.

Coupon Field Description	**Bit Size**	
B0	8	header of encoded coupon & first EC point
X1	256	x coordinate of the point
Y1	256	y coordinate of the point, optional (present only for B0 = 0x04)
Rnd/Rnd compressed	256 /16	random value
B1	8	header of second EC point, optional (present only for B0 = 0x40)
X2	256	x coordinate of second point, optional (present only for B0 = 0x40)
Y2	256	y coordinate of second point, optional (present only for B1 = 0x04)

Header Field Description (B0)	**Bit Size**	
Type	2	type of coupon, 0 for Schnorr, 1 for signcryption
Rnd	1	presence of random value
Rnd compression	1	compression of random value
Unused	1	zero
Uncompressed	1	EC point uncompressed
Compressed	1	EC point compressed
Parity	1	parity of y coordinate, used only if point is compressed

Header Field Description (B1)	**Bit Size**	
Unused	5	zero
Uncompressed	1	EC point uncompressed
Compressed	1	EC point compressed
Parity	1	parity of y coordinate, used only if point is compressed

chosen EC, $y^2 = x^3 + ax + b$, and one additional sign bit. This is also described in the SEC1 encoding guidelines [17].

The storage reduction that can be achieved through point compression is 32 bytes. An EC point is compressed from 65 bytes (1-byte header and 64 bytes for the X

& Y coordinates) to 33 bytes (1-byte header and 32 bytes for the X coordinate). Furthermore, two points are compressed for a signcryption scheme's coupon, since, two points are stored per coupon.

4.3.4 Index compression (i)

Index compression is used to compress the random value (denoted with y) in the coupon. Generally, the random value is a value of 32 bytes. Using a static master key (k) of 32 bytes and a counter value (ctr) of 2 bytes, the decompression shown in Equation (4.1) can be applied. It goes as follows: the counter value and the master key are concatenated and then hashed using the SHA256 hash algorithm. The resulting output is the random value used in the point multiplication. When index compression is used in the encoding of the coupon, the random value is replaced by the counter value. Note that the master key is only provisioned once on the device.

$$\text{Rnd} = \mathsf{H}(ctr \| k) \tag{4.1}$$

4.3.5 Coupon addition

Another proposed technique to reduce the storage requirements of the coupon is coupon addition. A larger set of coupons can be generated via distinct combinations of a smaller set of coupons. Moreover, two coupons can be added to each other using the equations shown in Equation (4.2). First, the random values are added via modular addition, which is the addition of the two values modulo the order N of the used elliptic curve group. Finally, the stored EC point(s), one for a Schnorr coupon and two for a signcryption coupon, are combined using point addition.

$$\begin{aligned} y_3 &= y_1 + y_2 \mod N \\ Y_3 &= Y_1 + Y_2 \\ k_3 &= k_1 + k_2 \text{ (signcryption)} \end{aligned} \tag{4.2}$$

The combinations are chosen using a counter value, shown in Table 4.2. The bit size of this counter value equals the length of the amount of coupons stored on the device. Moreover, the index of each bit in the counter represents the coupon of the stored set with the same index. The coupon is used if the respective bit in the counter value equals one. The counter value ranges from 1 to 2^n where n equals the amount of stored coupons. Using the example shown in Table 4.2, assume that eight coupons are stored on the device and the counter has reached the decimal value of 50. In this scenario, the second, fifth, and sixth bit of the counter are active. Thus, these three respective coupons are used in the combination. It requires two coupon additions. It follows that multiple or no coupon additions can take place, depending on the amount of active bits (ones) in the binary representation of the counter value.

Table 4.2: An example situation for combining coupons based on a counter value. In this example the counter has a value of 50, thus, coupons 2, 5, and 6 are added to each other.

Coupons	8	7	6	5	4	3	2	1	DEC
Counter	0	0	1	1	0	0	1	0	BIN (big-endian)
Result = $C_2 + C_5 + C_6$									

4.3.6 Comb method

The comb method for speeding up EC point multiplication is described in the Guide to Elliptic Curve Cryptography written by Hankerson et al. [41]. In comparison to the basic double-and-add algorithm, subsets of the bit-string representation of the random value is evaluated at once using the precomputed multiples of the base point (window). This comb method was extended by by Hedabou et al. in [42] to additionally provide countermeasures against side-channel analysis attacks; it is used in our work to implement the comb method through the Mbed TLS library. A more detailed description of the used comb method is provided in Section 2.3.3.

The window size can be chosen from $w = 2$ up to $w = 7$ in the standard Mbed TLS implementation. In our implementation, however, we extend this range to $w = 8$. Furthermore, the 'MBEDTLS_ECP_FIXED_POINT_OPTIM' macro of Mbed TLS is used to precompute the multiples of the base point. Then, these precomputed base point multiples can be used for an unlimited amount of point multiplications.

4.4 Implementation

Two separate implementations were created to evaluate the storage and computation methods described in Section 4.3. The first implementation evaluates the techniques based on pre-stored coupons (coupon encoding, memory encryption, point compression, index compression, and coupon addition); we refer to this as the Coupon implementation (Section 4.4.1). The second implementation generates coupons at run-time using the comb method; we refer to this as the Window implementation (Section 4.4.2).

Both implementations are programmed in C++ code and use Mbed OS as their operating system (version 5.8.1). Furthermore, all code is cross-compiled using the GNU Tools for Arm Embedded Processors (version 7-2017-q4-major). The platform used to test the implementations is the Maxim MAXREFDES100# health sensor platform [66]. It features a MAX32620 microcontroller, which uses an ARM Cortex-M4 with FPU CPU running at 48 MHz, a Bluetooth Low Energy (BLE) EM9301 communication chip, an additional 32 MB flash memory chip and a range of sensors. Moreover, the microcontroller has 2 MB of Flash Memory and 256 KB of SRAM.

The SECG secp256k1 Koblitz curve is used to produce the results for the evaluation. It provides a security level of 128 bits, which should be sufficient until 2028. Other elliptic curves could be used and might provide e.g. enhanced performance and/or security, however, the focus of this section is on the relative impact of the various storage and performance optimisation techniques, not on the exploration of different curves.

4.4.1 Coupon implementation

The Coupon implementation is used to evaluate point compression, index compression, coupon encoding, coupon addition, and memory encryption. To evaluate the different methods, the implementation is configurable through macros to only enable the required code for the respective configuration. The C/C++ preprocessor will recognise and expand the enabled macros in the code before it will pass the code onto the compiler.

The architecture of the Coupon implementation is presented in Figure 4.1. The Main class implements the functionality of requesting a coupon and generating a signature. In more detail, the following classes are implemented. A stored coupon, which is always encoded using the coupon encoding, can be requested via the Storage class. Furthermore, this class will also perform the decryption of the stored coupon if memory encryption is enabled. Thus, the output of the Storage class is the coupon in encoded format which can be decoded using the Coupon class. This Coupon class, therefore, implements the coupon decoding and all the decompression techniques, which are the index compression and EC point compression. Furthermore, the Main class can also retrieve a coupon from the Combinator class, which implements the coupon addition. The functional flow of the Combinator class is as follows: it retrieves the required coupon(s) for the next combination from the Storage class, then decodes and decompresses the coupon(s) via the Coupon class and finally adds the selected coupon(s) to each other and returns the resulting coupon. The decoded and decompressed coupons can be used by the Cipher class which implements the Schnorr or signcryption scheme to generate a signature on a message. In this implementation, all signatures are generated on the message “Hello World!”.

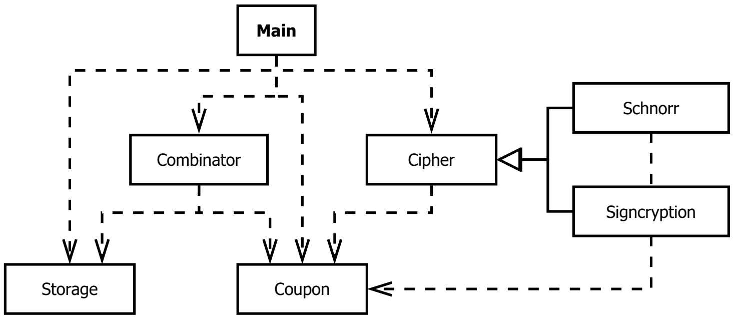


Figure 4.1: Simple representation of the Coupon implementation using a UML class diagram.

4.4.2 Window implementation

The Window implementation uses the Mbed TLS’s library for the point multiplication of the base point by the random value in the Schnorr signature scheme. The architecture of the implementation is presented in Figure 4.2. The Main class enables the functionality of the program. It uses the Generator class to create a coupon and the Schnorr Class to use this coupon to generate a signature on the message "Hello World!". The initialisation of the Generator consists of the precomputation of multiples of the base point (window initialisation), which has a configurable size, that will be used during the point multiplication based on the comb method. If the Main class requests a new coupon, the Generator class first generates a random value using a hash-based Random Number Generator (RNG), then it calculates the coupon using a point multiplication based on the comb method. Finally, it returns the calculated coupon.

The Window implementation only contains the Schnorr scheme, not the signcryption scheme. Though, the storage results of the signcryption scheme’s version of the Window implementation are estimated using the results of the Coupon implementation. The primary difference between the Schnorr and

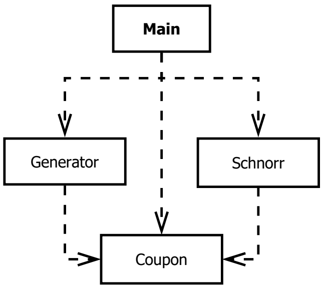


Figure 4.2: Simple representation of the Window implementation using a UML class diagram.

signcryption version of the Window implementation is the code implementing the signature schemes. Other differences are negligible in terms of storage size. The storage size of the Window implementation using the signcryption scheme requires an additional 5 kB of code and is estimated to be about 119 kB.

4.5 Results

The following results are evaluated for the Coupon implementation and the Window implementation: non-volatile memory (storage), performance, volatile memory (memory), and energy costs. To structure the tests, two different sets of configurations are used for the Coupon implementation: with and without coupon addition. These sets are called Addition and Basic, respectively. Both configurations are tested with selected combinations of techniques. The technique used in the combination is indicated by the following characters: ‘e’ for memory encryption, ‘c’ for point compression, ‘i’ for index compression, and ‘a’ for coupon addition.

The storage and static memory results are based on the analysis of the resulting binaries. The dynamic memory usage is monitored during execution of a signature generation by keeping track of the heap size. In Mbed OS, the heap memory section only grows when dynamically allocating and freeing memory. Consequently, the heap size at the end is the maximum amount of allocated memory that is required throughout the program. The performance results are measured using the available timer in Mbed OS. Next, the energy results are based on these performance results, since, energy (E) equals power consumption (P) times delay (t), $E = P \times t$. The power consumption value is obtained from the electrical characteristics section of the data sheet of the MCU. Also, the tested coupon set size is 8 and 365 for the Addition and

Basic configurations, respectively. Storage results using different set sizes are estimated from these fixed set sizes.

The Window implementation is evaluated with different window sizes: $w = 0$ and from $w = 2$ up to $w = 8$. A window size of $w = 0$ means that the precomputation of multiples of the base point is disabled.

4.5.1 Storage results

This section starts with a theoretical analysis of the coupon set size for each of the proposed combinations of techniques. Next, the storage requirement of the implementations is examined. Finally, these results are combined to estimate the amount of coupons that can be stored on a selection of storage sizes.

Theoretical computation of the storage size The impact of the proposed techniques on the coupon set size are compiled in equations that calculate the storage (s) required to have n coupons available for signatures. The Basic configuration is expressed by Equation (4.3) and the Addition configuration by Equation (4.4).

$$s = f \times n + g \tag{4.3}$$

$$s = f \times \log_2(n) + g \tag{4.4}$$

The parameters in both equations are defined by the proposed combination of techniques, displayed in Table 4.3. The first parameter f represents the size of the coupon which depends on the compression technique that is used. The second parameter g is the collection of static variables that are required for all the stored coupons.

Table 4.3: The parameters of Equation (4.3) and Equation (4.4) compiled for the six combinations (e: memory encryption, c: point compression, i: index compression).

	Schnorr		Signcryption	
	f (kB)	g (kB)	f (kB)	g (kB)
e	0.097	0.048	0.162	0.048
c	0.065	0	0.098	0
i	0.067	0.032	0.132	0.032
ei	0.067	0.08	0.132	0.08
ec	0.065	0.048	0.098	0.048
eic	0.035	0.08	0.068	0.08

The coupon has the largest size when only memory encryption (e) is used. This is 97 and 162 bytes for Schnorr and signcryption, respectively. Point compression (c) has a larger effect on the signcryption coupon than on the Schnorr coupon, because the signcryption coupon stores two EC points. The effect of index compression (i) is the same for the Schnorr and the signcryption coupon. Furthermore, index compression and memory encryption require a small portion of static variables, but these are negligible in comparison to the storage size of the coupons and the static size of the implementation (without coupons).

Static size The storage requirements of the different implementations without coupons are shown in Table 4.4. As a reference, the size of the base implementation is also given, i.e. the size of the implementation without optimisation methods. Using these values, we have distilled the storage requirements for each optimisation. Memory encryption (e) requires a static size of about 3 kB in the Schnorr configuration, while in the signcryption configuration, it requires only 1 kB. The reason is that the AES cipher is already present for the signcryption scheme and does not need to be included again for the memory encryption. Point compression (c) adds around 2 kB of code for both the Schnorr and signcryption scheme. It requires the algorithms to calculate the corresponding coordinate of the compressed point. Index compression (i) adds a negligible amount of code because it uses the SHA256 hashing algorithm that is already used for the signature generation in both the Schnorr and signcryption scheme. Coupon addition (last column in Table 4.4) adds about 7 kB of code.

Table 4.4: Static size (in kB) of the implementation without coupons.

Schnorr	Basic	Addition
base	105.6	111.9
e	108.6	113.8
ec	110.3	115.9
ei	108.5	113.9
eic	111.2	116.0
Signcryption		
base	110.9	117.6
e	111.9	117.6
ec	114.2	119.8
ei	111.8	117.7
eic	114.2	119.9

Coupon set size In this paragraph, the amount of signatures versus the coupon set size is explored, which involves the calculation of the amount of coupons that can be stored, shown in Table 4.5. The results from this table take into account the static size of the implementation. Equations (4.3) and (4.4) are used to calculate the amount of coupons that can be stored for a specific total size of available storage. Thus, the resulting value is the amount of coupons that can be used for the generation of consecutive signatures. The chosen total sizes of available storage are the following: (1) the size of the Window implementation (114 kB or 119 kB), which can generate an infinite amount of coupons; (2) the size of commonly used flash storage (128 kB); (3) the maximum storage available on the health sensor device's MCU (2048 kB). Note that, for the Addition configurations, the number of generated coupons is extremely high and therefore only added to the table for storage capacity (1) and (2).

Table 4.5: The amount of coupons that can be used for consecutive generation of signatures. It is calculated by subtracting the available space by the respective implementation size and using the remaining free space for coupon storage.

	Amount of usable coupons				
Storage (kB)	**basic**	**e**	**ec**	**ei**	**eic**
Schnorr					
Basic					
114	86	55	56	80	77
128	230	199	271	289	477
2048	20024	19993	29810	28946	55334
Addition					
114	2.1E+06	1.0E+00	0	0	0
128	4.7E+49	4.5E+43	4.9E+55	8.2E+62	2.2E+102
Signcryption					
Basic					
119	49	43	48	53	69
128	105	99	140	122	201
2048	11957	11950	19732	14667	28437
Addition					
119	2.6E+02	2.6E+02	0	5.1E+02	0
128	9.2E+18	9.2E+18	9.7E+24	1.5E+23	1.7E+35

The basic configurations express as expected a linear relationship between the amount of coupons that can be stored and the available storage. On the other hand, the amount of coupons for the addition configurations grow exponentially, because, 2^n coupons can be formed for n coupons. The compression techniques display a similar behaviour. These techniques have an exponential impact on the Addition configurations but only a linear impact on the Basic configurations.

4.5.2 Performance results

The performance measurements are split into four paragraphs. The first three involve the coupon initialisation or generation using the proposed storage methods, and the final paragraph concerns the timing results of the signature generation assuming that a set of coupons are available.

Coupon initialisation for compression techniques without coupon addition This paragraph discusses the results of the precomputation-based implementations that do not use coupon addition and are referred to as Basic implementations up to now. First, the performance results of the Schnorr coupon initialisation are discussed for each of the selected storage method combinations and displayed in Figure 4.3. The results are split into two graphs with different scales because of the large timing differences. Point compression (c) is notably more compute-intensive than the other storage methods. It requires about 29 ms to decompress one EC point. This is vastly more than the timing results of the encrypted storage and index compression (i) methods. Memory encryption (e) adds an overhead of about 0.5 ms and index compression around 0.1 ms.

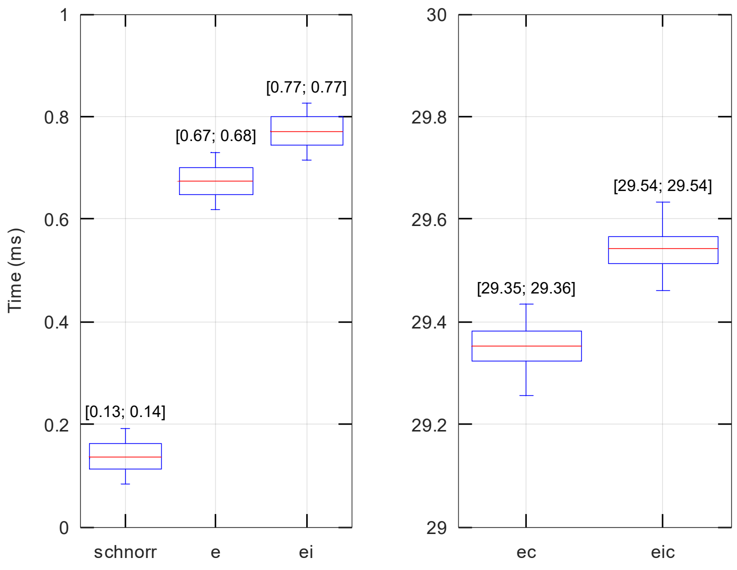


Figure 4.3: Performance results of the Schnorr coupon initialisation of the Coupon implementation using the different storage methods without coupon addition.

Secondly, the signcryption-based configurations are evaluated, as shown in Figure 4.4. These results are similar to the Schnorr-based configurations except for the point compression (c) method. The performance of the point compression method is halved because two EC points instead of one need to be decompressed. It requires about 57 ms to calculate two EC points for a signcryption coupon. Memory encryption (e) and index compression (i) require respectively around 0.6 ms and 0.1 ms. Furthermore, the decryption time is slightly higher for Schnorr than for signcryption; this is probably due to the fact that the signcryption coupon is larger than the Schnorr coupon.

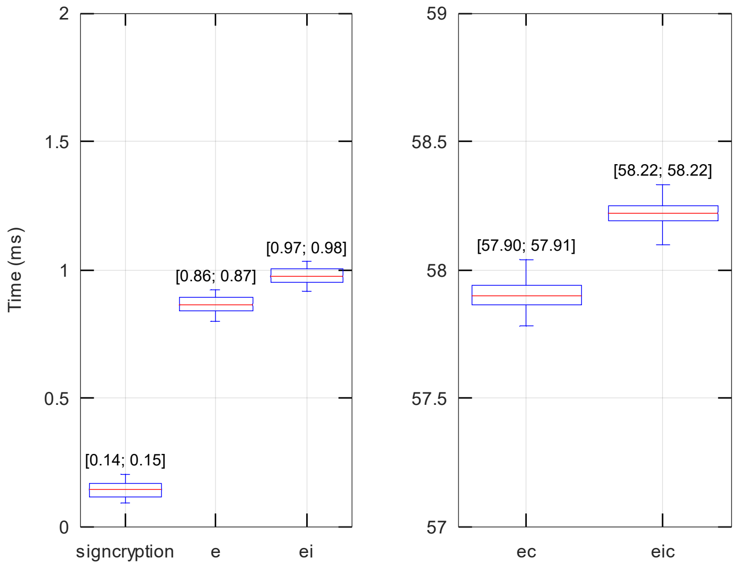


Figure 4.4: Performance results of the signcryption coupon initialisation of the Coupon implementation using the different storage methods without coupon addition.

Coupon initialisation for coupon addition This paragraph concerns the performance implication of the coupon addition method; these implementations are referred to as Addition configurations. The results largely depend on the coupon combination that is chosen and are therefore expressed as such. The amount of coupon additions performed in the coupon combination is determined by the Hamming weight of the counter value in binary format. A coupon addition must be performed for each additional coupon in the combination. The graph in this paragraph is, therefore, expressed as a function of the Hamming weight (amount of coupon additions) of the combinations, shown in Figure 4.5.

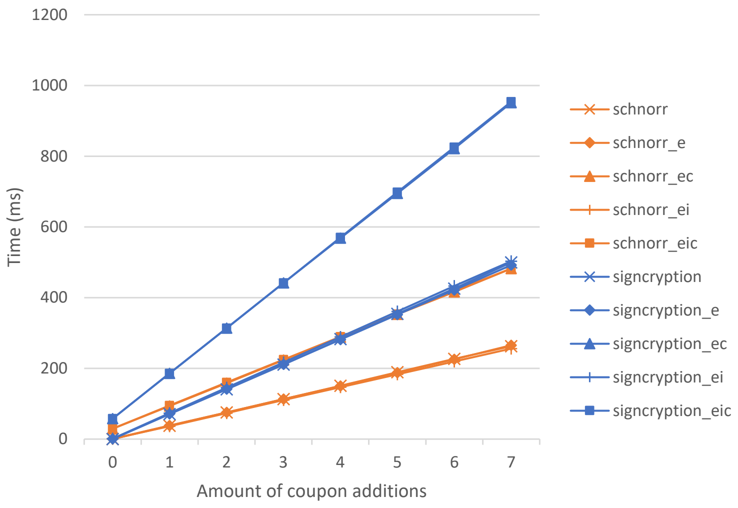


Figure 4.5: Performance results of Schnorr and signcryption coupon initialisation as a function of the amount of coupon additions.

The timing results of the signcryption-based configurations are nearly twice as slow as the Schnorr-based configurations. This is because the signcryption coupon requires two point additions instead of one. Furthermore, memory encryption (e) and index compression (i) have a negligible effect on the performance of the coupon additions. Point compression (c), on the other hand, is more compute-intensive and almost doubles the timing results. The amount of coupon additions expresses a similar linear behaviour, since, each coupon addition requires a fixed amount of time.

Coupon generation using the comb method The results of the coupon initialisation methods using the Window implementation based on the Schnorr scheme are shown in Figure 4.6. Two sets of measurements were performed: the window initialisation (Initialisation) and the Coupon initialisation (Point multiplication). The initialisation of the window by the health sensor was added to evaluate the potential performance impact. This window/multiples of the base point, however, could also be stored in the device like it is done in the Coupon implementation. The window size can be configured from a size of $w = 2$ till $w = 8$. As seen in the graph, a larger size would not be beneficial for the performance. In terms of timing results, the performance of the coupon initialisation can be increased by increasing the window size. However, the performance gain decreases proportional with the size of the window. Moreover, in the Mbed TLS library, the window size is automatically configured to $w = 5$, which could be considered an optimal point. In short, a point multiplication requires at minimum 230 ms of computation. On the other hand, the time required to initialise the window increases exponentially, and, it intersects with the coupon initialisation line at about $w = 3$.

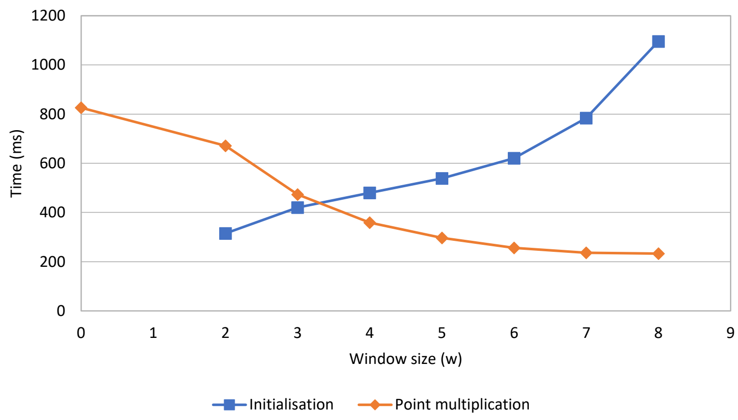


Figure 4.6: Performance results of the Schnorr implementation using the comb method.

Signature generation The last timing results are compiled from the signature generation step, displayed in Figure 4.7. These results are independent from the storage methods, since, a fully initialised coupon is always used to generate a signature. The Schnorr and signcryption signatures require about 2.6 and 3.5 ms, respectively. The signcryption results are slower because of the additional encryption step.

4.5.3 Memory results

The memory results of all implementations are presented in Figure 4.8. The memory results were compiled via the analysis of the static (stack) and dynamic (heap) memory. The stack is represented by the amount of data that is reserved for statically allocated data. The memory required by Mbed OS has been subtracted. All implementations express a similar memory usage of about 3 kB of heap and 270 B or 9 kB of stack, depending on if the AES algorithm is used. The tables of the AES algorithm are stored in memory, however, the Mbed TLS library can also be configured to store these tables in storage. In comparison to the coupon implementation, the Window implementation requires

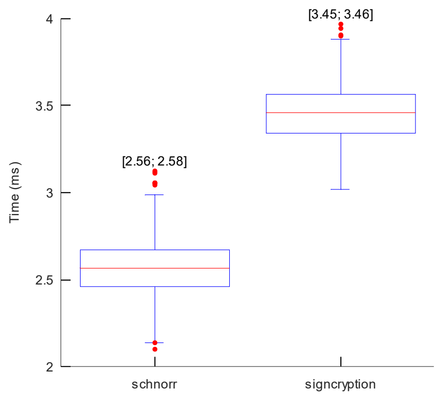


Figure 4.7: Performance results of Schnorr and signcryption signature generation using their respective initialised coupon.

more heap memory depending on the window size. It requires about 1.7 times more heap memory if we consider a window size of $w = 5$.

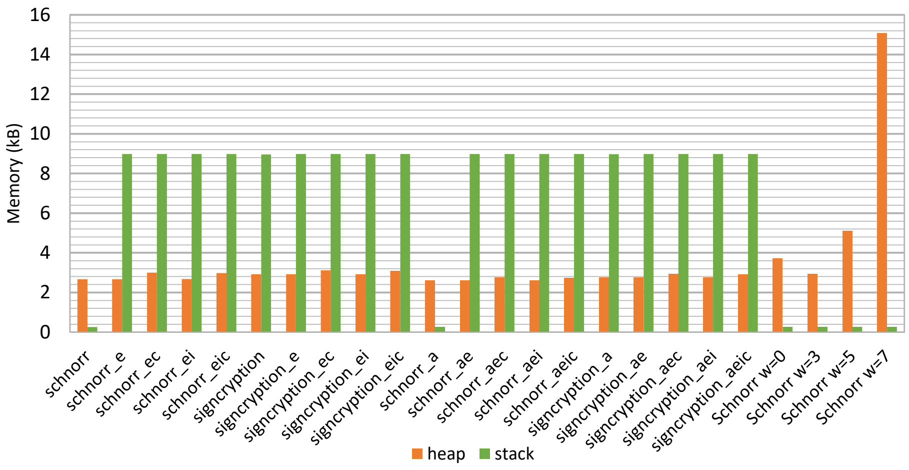


Figure 4.8: Memory usage of the Schnorr and signcryption scheme using the different optimisations (e: memory encryption, c: point compression, i: index compression, a: coupon addition, w: window width).

4.5.4 Energy results

The computation and the communication energy consumption of the implementations are estimated using an analytical approach, shown in Figure 4.9. In terms of communication, we consider only the size of the signature. The energy results are calculated using the performance results and the power consumption parameters available in the datasheets of the MAX3260 MCU and the EM9301 network interface chip.

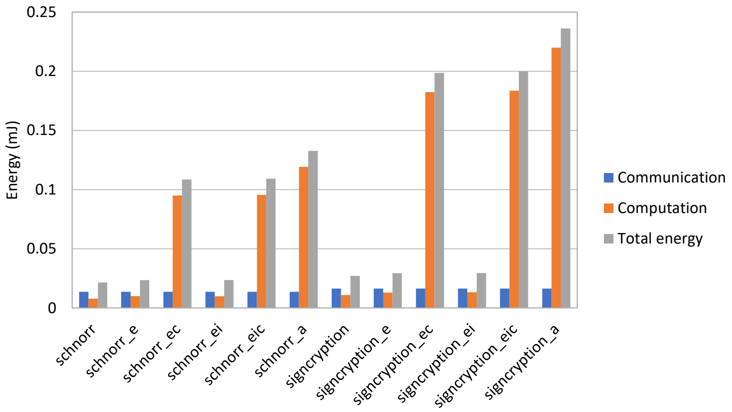


Figure 4.9: Energy usage of the Schnorr and signcryption scheme using the different optimisations (e: memory encryption, c: point compression, i: index compression, a: coupon addition). A signature generation with one coupon addition is used to represent the addition optimisation.

The results show that the energy cost of the communication and the computation are similar when no optimisation techniques are used. They take up respectively about 60% and 40% of the total energy. Furthermore, the memory encryption and index compression do not have a large impact on the energy results, while, the point compression and coupon addition require much more computation power. They require respectively about 10 and 12 times more energy. However, the generation of a coupon using the comb method ($w = 5$) requires around 0.9 mJ which is about 4 to 112 times more than the energy cost of a coupon initialisation using the coupon optimisations.

4.6 Discussion

A challenge of the coupon based signature generation is the amount of precomputed coupons that can be used. It is analysed using the **storage results**. Using the measured results and the compiled equations, the amount of usable coupons, i.e. the coupons that can be used to generate a signature, were calculated. In this regard, the point compression and index compression expressed a linear impact, while, the coupon addition showed a exponential impact. In terms of static storage, index compression requires a negligible overhead, and, point compression and coupon addition use respectively about 2 kB and 7 kB of code. The memory encryption technique requires respectively about 3 kB and 1 kB of code for the Schnorr and signcryption scheme.

Two different sets of **timing results** are examined: coupon initialisation and signature generation. For the coupon initialisation, three different sets of configurations are analysed. The first is the Basic configuration, which does not include the coupon addition technique. Memory encryption and index compression have a relatively small impact, up to 1 ms, while point compression requires a considerable amount of computation, 29 ms and 58 ms for Schnorr and signcryption, respectively. The second configuration uses the coupon addition. This method also requires a considerable amount of computation for each addition, about 35 ms for the Schnorr coupon and 70 ms for the signcryption coupon. The amount of additions depends on the combination of coupons chosen via the active bits in the counter value. Thus, the total amount of time required scales proportionally with the Hamming weight of the counter value. The third configuration is about the comb method. The performance of the point multiplication for generating a coupon can be enhanced up to a window size of $w = 7$, resulting in around 230 ms for generating a coupon. In comparison to the comb method, the proposed techniques have a lower performance impact on the initialisation of the coupon, except for those with a higher number of coupon additions. Nevertheless, this can be avoided by using only smaller combinations, i.e. combinations with a limited amount of coupon additions. The last set of timing results is provided for the signature generation process. The Schnorr- and signcryption-based configurations take about 2.5 and 3.5 ms, respectively. Only the impact of memory encryption and index compression are relatively small in comparison to the signature generation process. Point compression and coupon addition on the other hand require much more computation time.

The different proposed techniques do not have a large impact on the **memory results**. About 3 kB of heap and 270 B or 9 kB of stack is required. If the AES algorithm is used, the implementation requires an additional 8 kB of stack memory to store its tables. In comparison to the Coupon implementation, the Window implementation requires around 1.7 times more heap memory if a window size of $w = 5$ is assumed.

The **energy results** are based on the performance results, and, both the computation and communication energy cost are estimated. Note that the optimisation techniques only have an impact on the computation, because, only the signature size of the Schnorr and signcryption scheme have an influence on the communication. The computation and communication energy cost are similar and take up respectively about 60% and 40% of the total energy if no optimisation techniques are used. The memory encryption and index compression do not have a large impact on the computation energy consumption, while, point compression and coupon addition require respectively around 10 and 12 times more energy. In comparison to the Window implementation, the coupon optimisations require about 4 to 112 times less energy.

In summary, the results show that all proposed optimisations should be used to maximise the available coupons. Though, the use of point compression in low latency and energy-efficient applications is less useful. The reason is that this technique shows a similar performance impact than the coupon addition optimisation, while, it has a far lower impact on the available coupons. However, the coupon addition optimisation does not scale well in larger combinations of coupons in terms of performance and energy cost.

4.7 Conclusion

A total of six techniques to enhance the practical usage of Schnorr-based signature and signcryption schemes are evaluated. The evaluation of these methods is done on two different implementations, which we call the Coupon and the Window implementation. The Coupon implementation uses the technique described by Winderickx et al. [123] for the precomputation of coupons, and, it implements both the Schnorr and signcryption scheme. Moreover, it can be configured to a selection of the following five techniques: coupon encoding, memory encryption, point compression, index compression, and coupon addition. In this implementation, the coupon encoding technique is always used for storing the coupons to facilitate the coupon processing. The other methods are evaluated in predefined combinations. The combinations are divided in two configuration sets: with (Addition) and without (Basic) coupon addition. The following five technique combinations are used: (1) no optimisation, (2) memory encryption, (3+4) memory encryption and either index compression or point compression, and (5) all of them combined. The Window implementation on the other hand is used to evaluate the comb method which enhances the performance of EC point multiplication. Only the Schnorr-based scheme is tested using this method. Nevertheless, the implementation size of the signcryption-based version is estimated. All tests are performed on the MAXREFDES100# health sensor device. For both implementations, four parameters are examined: storage, performance, memory, and energy.

The results show that all proposed techniques enhance the practical usage of the Schnorr and signcryption scheme. The most promising technique is coupon addition, where combinations of coupons can be used to exponentially increase the amount of usable coupons. Though, it does not scale well in terms of performance. On the other hand, point compression has a similar impact in terms of performance and energy as coupon addition, but, it only provides a linear impact in terms of usable coupons that can be stored. Next, index compression expresses a negligible overhead for all measured parameters. The use of memory encryption technique should be mandatory as it secures the coupons stored on the devices. Moreover, it had a small impact in terms of performance and energy results. In terms of storage and memory, the memory encryption adds up to 3 kB of code if an AES implementation is not already present, and about 8 kB of memory usage if the AES tables are stored in memory. Overall, the impact of all the proposed techniques have a smaller impact in terms of storage, memory, performance, and energy in comparison to the reference point multiplication optimisation (comb method).

Chapter 5

In-depth energy analysis of secure IoT computation and communication

This chapter provides an in-depth analysis of the complete energy cost, i.e. the computation and communication aspect, of security protocols deployed over a wireless communication link. The energy consumption was estimated using a granular approach. Furthermore, an equation is given that calculates the minimal time period of a secure communication session in order to minimise the energy impact of the session's setup phase and thus minimise the overall average power consumption. First, the scientific contribution of this chapter is described in Section 5.1. Next, related work is discussed in Section 5.2. In Section 5.3, the granular approach to evaluate the energy cost of the security computation and communication is described. Then, the results of the basic operations that are required for the granular approach are presented in Section 5.4. The energy costs of providing an end-to-end secure communication channel, on the one hand, and digital signatures, on the other hand, are evaluated in respectively Section 5.5 and Section 5.6. Next, our approach to calculate the minimal session lifetime is presented in Section 5.7. Finally, the chapter is concluded in Section 5.8.

5.1 Introduction

In order to quantify the impact of security provisions in IoT devices, this chapter concentrates on the energy consumption of security algorithms and protocols for end-to-end security and digital signatures. We divide the energy cost of a device into three parts: communication, application and security. The communication cost involves the transport of packets from the IoT device to the remote end user and vice versa. The application cost involves the operation of sensors and related data processing algorithms. Finally, the security cost consists of the execution of security algorithms to provide confidentiality and authentication. While the application cost is specific to the use case, the communication and security cost can be generalised. A lot of research has, therefore, gone into the evaluation of one of these two costs. However, security and communication costs are tightly intertwined. Security is required to protect wireless communication, but communication is required to enable security. In this chapter, we explore both the communication and computation cost of providing and using an end-to-end secured communication channel in an IoT setting using a granular approach. It is not the intention of this chapter to provide accurate energy measurement results. Rather, the focus is on estimating the relative impact of end-to-end security on the total energy consumption of the application.

A security session typically contains a setup and a runtime phase. During the setup, entities can be authenticated, and, shared cryptographic material is established. This is used during the runtime phase to encrypt and/or authenticate the data. However, the runtime phase has a finite lifetime. After this period, a new session must be established. The longer a session is maintained, the higher the risk of the security session getting compromised. The most basic method to determine the maximum duration of a session is to calculate the maximum amount of data that can be encrypted by an encryption key. Another method is to estimate the risk of a successful side-channel attack in order to define the maximum lifetime of a session based on the number of traces an attacker can collect. The number of side-channel traces measured with the same key is typically proportional to the chance of a successful attack. Hence, in order prevent a successful attack, the lifetime of a session should be limited. While these two methods put an upper bound on the session duration, in this chapter, we introduce a lower bound on the session lifetime in order to minimise the impact of the setup phase on the overall energy consumption. This way, the lower and upper bound of the session duration is defined and a trade-off between the average power consumption and the side-channel resistance can be made.

We summarise the contributions of our work as follows:

- We are the first to perform an in-depth analysis of both the communication and the computation energy cost to achieve end-to-end security and digital signatures for a broad range of IoT platforms, security algorithms and protocols, and wireless communication standards. Three different methods for setting up an end-to-end secured channel and five different modes of operation for the AES algorithm are evaluated.
- We introduce the calculation of a lower bound on the duration of the runtime phase of a security session in order to minimise the overall average power consumption of the session. We evaluate this metric for two different application scenarios, one with a relatively high data rate and one with a relatively low data rate.

My contribution to the work presented in this chapter was the in-depth implementation and analysis of the computation and communication requirements of the considered algorithms and protocols. I also proposed to evaluate the impact of the session duration on the energy consumption.

5.2 Related Work

Research on the implementation of cryptographic algorithms on MCUs is primarily focused on performance, memory usage and/or energy consumption [28, 58]. For example, the paper published by Ledwaba et al. [58] analyses the cost of cryptographic software in recent end-point devices. The authors look at the performance of the AES-CTR, SHA256 and ECDSA algorithms. A comparison is made between the different types of ARM Cortex-M devices.

A diverse set of wireless network protocols is analysed by Sen et al. [99]. The authors evaluate the energy consumption of the network protocols and discuss their security strength.

Although the aforementioned comparisons are useful for practitioners and researchers in the field, they do not give a complete view on the energy requirements of cryptographic protocols providing end-to-end secure communication and digital signatures. Our work performs an in-depth and complete analysis, taking into account the energy for cryptographic computations as well as the energy that is necessary to transmit messages between the IoT device and the end user. Trappe et al. [113] perform a study that is similar to ours. The authors do measurements on the MPS430g2553 16-bit MCU and the CC1150 Multichannel RF Transmitter. However, their work is limited to an experiment that transmits 5 B of data, without taking into account the setup/handshake phase that needs to be executed at the start of each communication session. In our work, we evaluate the energy consumption of three different implementation platforms, three different wireless communication standards, and several algorithms for both the setup and the runtime phase of a secure communication session. In addition, we analyse digital signatures. We use these results to determine a lower bound on the session duration for two different application scenarios, one in which 33 kB is transmitted every 10 seconds (wearable healthcare device) and one in which 224 B is transmitted every half an hour (weather station).

5.3 Our approach

The computation and communication energy cost of algorithms and protocols that are required to provide end-to-end security and digital signatures are analysed using a granular approach. This means that basic operations will be identified for each considered algorithm. The total energy cost of an algorithm is, then, equal to the sum of the energy costs of the required basic operations.

The computation energy cost (E_{comp}) of the basic operations is based on their execution time (t), expressed in number of cycles, in accordance to Equation (5.1). The power (P_{comp}) characteristics are determined using the values available in the data sheet of the device, which present the average power consumption per cycle.

$$E_{comp}(t) = P_{comp} \times t \tag{5.1}$$

The communication energy cost ($E_{TX/RX}$) of the cryptographic protocols is based on the size of the basic messages that need to be communicated. Via the throughput (Th) and power consumption ($P_{TX/RX}$) of the wireless network, the energy required to send these basic messages of size (M) can be determined via Equation (5.2).

$$E_{TX/RX}(M) = \frac{P_{TX/RX}}{Th} \times M \tag{5.2}$$

Only considering the basic elements and the featured electrical parameters of the computation and communication provides only a rough energy cost estimation. Note that it should provide sufficient accuracy, as the goal is to estimate the relative impact of end-to-end security in terms of communication and computation. Though, other contributing factors like wireless interference and etc. might negatively impact the energy cost of the communication. Thus, we assume that more accurate estimations will have a small impact on the relative share between the computational and the communicational energy cost. Next, The different computation platforms and communication standards are first discussed separately, as a MCU should not be limited to a certain communication standard. After the description, we only consider the combinations that are available according to the used platforms in our results, so that the amount of results are manageable.

5.3.1 Evaluation platforms

The performance of the basic operations is measured on three different microcontroller platforms. These platforms are chosen to cover a range of different sizes of flash program memory, data memory, and operating frequency. They are described below and more detailed specifications are given in Table 5.1. The table also indicates the P_{comp} value for each platform, which is necessary for the calculation of the computation energy cost according to Equation (5.1).

nuc The NUCLEO-L073RZ is a STM32 Nucleo-64 Development Board of STMicroelectronics [105]. It features the STM32L073RZT6 32 MHz ARM Cortex-M0+ microcontroller with 192 KB flash memory and 20 KB RAM.

msp The TI SimpleLink MSP-EXP432P401R LaunchPad development kit [110] features the MSP432P401R 48 MHz ARM Cortex-M4F microcontroller with 256 KB flash and 64 KB RAM.

max The MAXREFDES#100 health sensor platform [66] uses the MAX32620 96 MHz ARM Cortex-M4F microcontroller with 2 MB flash and 256 KB RAM. It has a wide range of sensors, like a human body temperature sensor and a heart rate sensor.

Table 5.1: Technical specifications of the considered evaluation platforms.

	max	***msp***	***nuc***
MCU	MAX32620	MSP432P401R	STM32L073RZT6
Frequency (MHz)	96	48	32
Flash (KB)	2000	256	192
RAM (KB)	256	64	20
Voltage (V)	1.2	3	3
Current (mA)	9.79	7.70	6.65
Power (mW) - P_{comp}	19.95	23.10	11.75

5.3.2 Wireless networks

IoT applications can require a wireless communication range of meters to kilometres. To cover both short-range and long-range applications, we consider the following three protocols:

- Bluetooth Low Energy (BLE),
- Wi-Fi,
- Long Range Wide Area Network (LoRaWAN).

The characteristics of LoRaWAN are provided for two configurations: slowest mode (SF 12) and fastest mode (SF 7). The specifications for all protocols are given in Table 5.2. All parameters except for the throughput of the BLE network and the energy efficiency are collected from the data sheets of the RF chip that is used to enable the respective communication protocol. Moreover, the values used are primarily from the data sheets' features section. To use the LoRaWAN network, the *nuc* platform is expanded with a Semtech SX1272MB2xAS LoRa extension board [98]. Wi-Fi is provided to the *msp* platform by using the SimpleLink Wi-Fi CC3120 wireless network processor BoosterPack plug-in module. Finally, a BLE compatible RF chip is already present on the *max* platform. It uses the EM9301 BLE controller [35] that supports BLE version 4.1.

The speed at which a wireless network can communicate symbols is not the same as the speed at which data can be sent. The difference is in the overhead of the network protocols in the data link layer. In order to perform an accurate analysis, the actual throughput may have to be calculated. Only the data sheet of the BLE chip does not report on the throughput metric. The LoRa bit rate could also be calculated using the Shannon-Hartley theorem [20]. Additionally, the application needs to take the Fair Policy Access, which limits the amount of data an application is allowed to send [112], into account in a practical LoRa setting. The throughput of the BLE network is calculated using the specifications of the Data link and Physical layer. In BLE version 4.1 [10], a typical message has 14 B of headers and a maximum payload size of 27 B. Furthermore, there is an inter-frame space of 150 μs, i.e. the required interval between two consecutive packets. Taking this into account, the parameters that are necessary to compute the communication energy cost in Equation (5.2) are indicated in Table 5.2.

Table 5.2: Technical specifications of the considered wireless protocols.

	BLE (4.1)	**Wi-Fi**	**LoRA (SF 12)**	**LoRA (SF 7)**
Electrical characteristics				
Evaluation platform	*max*	*msp*	*nuc*	*nuc*
RF chip	EM9301	CC3120	SX1272	SX1272
Bandwidth (MHz)	1	20	0.125	0.25
Symbol rate (kbps)	1000	54000	0.366	13.7
Throughput (kbps) - Th	305	16000	0.293	10.94
Range (km)	0.1	0.25	14	2
Voltage (V)	2.5	3.6	3.3	3.3
TX				
Current (mA)	12	59	28	28
Power (mW) - P_{TX}	30	212.4	92.4	92.4
Energy efficiency ($\mu J/B$) - $8P_{TX}/Th$	**0.79**	**0.11**	**2523**	**67.58**
RX				
Current (mA)	13	229	11.2	11.2
Power (mW) - P_{RX}	32.5	824	37.0	37.0
Energy efficiency ($\mu J/B$) - $8P_{RX}/Th$	**0.852**	**0.412**	**1009.3**	**27.034**
Idle				
Current (mA)	0.009	0.690	1.4	1.4
Power (mW)	0.023	2.484	4.62	4.62

5.3.3 Computation

In terms of public-key algorithms, only those based on Elliptic Curve Cryptography (ECC) are considered. These algorithms can be divided into the elliptic curve (EC) arithmetic operations that are used. The following four operations are considered: point addition (PA), point doubling (PD), point multiplication (PM), and fixed-point multiplication (PMG). A distinction is made between a random point multiplication and a fixed-point multiplication, because, the comb method optimisation can be used for the fixed-point multiplication. This optimisation requires the pre-calculation of a set of EC points to increase the performance of the multiplication process, but, it is only efficient if a point is used multiple times. This is typically the case for the base point (G) of the chosen elliptic curve.

Hash functions and symmetric-key ciphers typically have a much higher performance than public-key based algorithms. For this reason, these algorithms were not divided into basic operations but are considered basic operations themselves. The following hash functions are considered: SHA256 and SHA3-256. For the symmetric-key ciphers, AES is used in the following five modes of operation: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Counter (CTR), Counter with CBC-MAC (CCM), and Galois/Counter Mode (GCM). A key size of 128 bits is used for the AES algorithm.

The performance of all identified basic operations is measured on the three platforms. 50 time measurements are done for each basic operation using the platforms' available timer. Moreover, the AES cipher operation is an encryption on 256 Bytes of data. We have chosen a multiple of the AES block size, because, longer time periods ensure less influence of potential timing inaccuracies like an early start and late end. For the hash function, the maximum input size of the respective algorithm for one round is chosen as follows: 55 B for SHA256 and 135 B for SHA3-256. The total available internal state size is not used for SHA256 and SHA3-256, as we take into account the minimal padding or suffix that is required for the last block of input data. Note that the most optimal scenario, i.e. the maximum amount of input data to fill up the internal state completely, is used for each of the operations. Bar charts with error bars and the 95% percentile are used to represent the results and the spread. The energy cost is calculated using Equation (5.1). Additionally, the energy results of the AES ciphers and hash algorithms are divided by their message input size to calculate the energy required per byte.

All basic operations are implemented using software libraries and cross-compiled with the GNU Tools for ARM Embedded Processors version 6-2017-q2-update. Furthermore, the compiler is configured to optimise for size (-Os). The RELIC-toolkit library [2] is used to implement the EC arithmetic and the SHA256 hash function. We use the SECG K-256 prime elliptic curve, "BASIC;COMBA;COMBA;MONTY;MONTY;SLIDE" configuration for the prime field arithmetic, and "PROJC;LWNAF;COMBS;INTER" configuration for the prime elliptic curve arithmetic. For more information on how to configure RELIC and other examples that use it, we refer to the wiki [2] and to an example [47]. The AES ciphers are implemented using Mbed TLS [4] and SHA3 using wolfCrypt [127]. We use the SHA3-256 hash function as specified in FIPS PUB 202 [34].

5.3.4 Communication

Public-key based algorithms typically need to communicate a selection of the following four basic objects: an EC point, a signature, a random number, and/or a certificate. The curve we use throughout the chapter is SECG K-256. For this curve, an uncompressed point can be compiled in 65 Bytes using the SEC1 encoding. The size of a signature depends on the algorithm used, e.g. the ECDSA scheme produces a 64-Byte signature. A typical random number uses 32 Bytes. Finally, the size of a certificate depends on the type. For example, a SECG K-256 based X.509 certificate requires 544 Bytes. This value was measured by generating a basic certificate using the OpenSSL command line interface.

5.4 Energy Results for the Basic Operations

The energy cost of the elliptic curve arithmetic is presented in Figure 5.1. The use of the comb method for the fixed point multiplication explains the enhanced performance in comparison to the general point multiplication. Furthermore, the addition and doubling operations pale in comparison to the point multiplication.

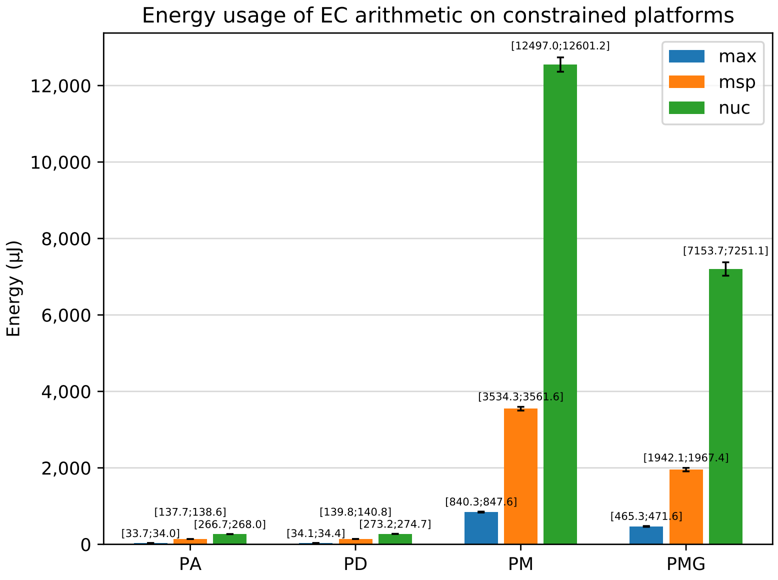


Figure 5.1: Energy cost of elliptic curve arithmetic on the three considered platforms (PA: point addition, PD: point doubling, PM: point multiplication, PMG: fixed-Point multiplication).

The results of the SHA256 and SHA3-256 hash function are presented in Figure 5.2. Note the different scale on the vertical axis. In terms of energy per byte, SHA256 is about 40% more energy efficient than SHA3. In terms of performance, one round of SHA3 is much slower than one round of SHA256, but, it can input more bytes per round. SHA256 can input 55 B while SHA3 can handle 135 B of data. On the max platform, one round takes around 108 μs and 438 μs for respectively SHA256 and SHA3. The advantage of SHA3 is the strong security rationale behind it. We refer to Chapter 2 for more information.

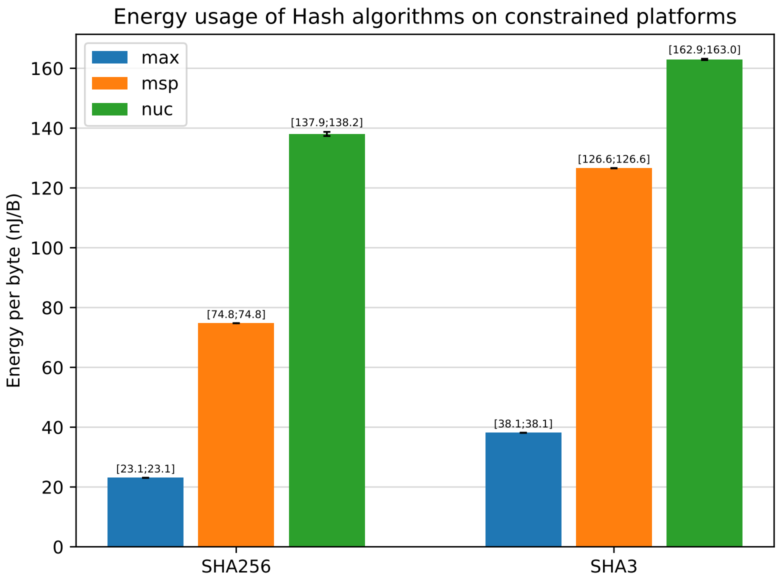


Figure 5.2: Energy cost per byte of the SHA256 and SHA3-256 hash functions on the three considered platforms.

The results of the AES cipher and operation modes are shown in Figure 5.3. The simplest block cipher mode, ECB, is, as expected, the most efficient with about 21 nJ/B and 77 nJ/B for respectively the max and nuc platform. The CBC and CTR mode express similar but slightly higher energy costs. The AEAD cipher modes, GCM and CCM, require about double the energy per byte in comparison to the basic modes. This can be attributed to the authentication operation that is additionally provided.

5.5 Energy Results for End-to-end Security

A secure communication channel is the channel used between two parties (client and server) to provide end-to-end security guarantees as shown in Figure 5.4. Throughout this section, the constrained device is used as the server and a

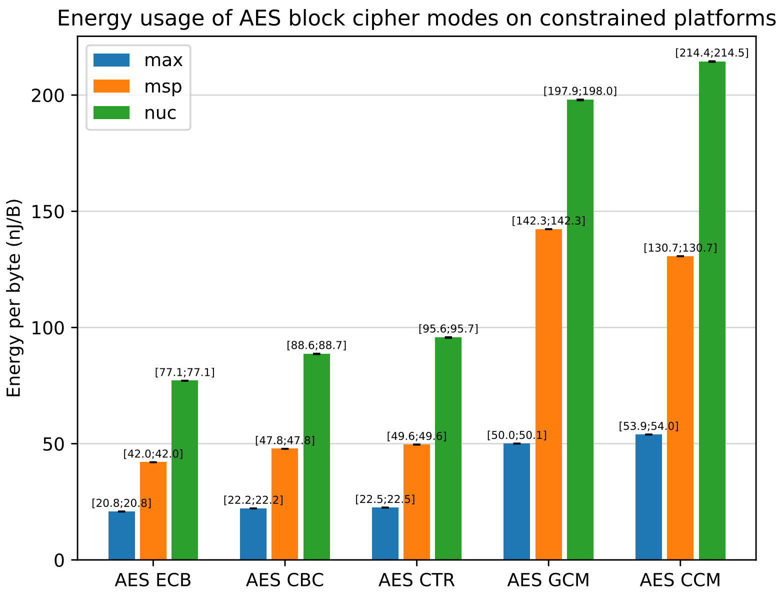


Figure 5.3: Energy cost per byte of AES with modes of operation on the three considered platforms.

remote party as a client. Two methods of providing security are considered in this chapter: via encryption and via signatures. More details on encryption, signatures, and end-to-end security can be found in Chapter 2.

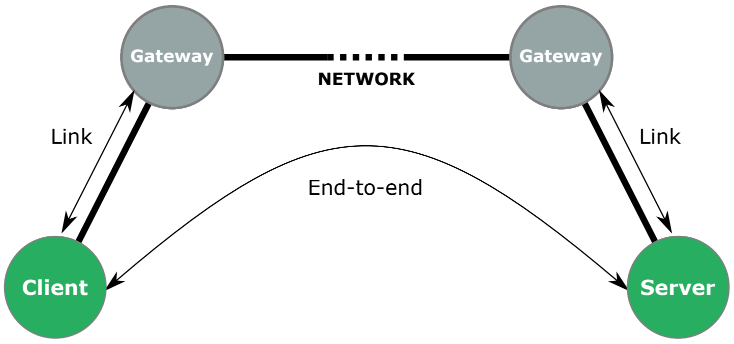


Figure 5.4: Schematic representation of an end-to-end secured connection.

For applications that use encryption for end-to-end security, three methods of establishing a shared key are analysed in this chapter: pre-shared key (PSK), pre-shared public key (PSPK), and pre-shared trusted third party public key (PSTTPPK). PSK implies that a shared encryption key is already present on both communicating entities. This is typically done via an out-of-band method. Authentication of the entities is, then, implicitly assumed through the use of this shared key. PSPK means that the public key of the remote party is pre-configured, which is also done via an out-of-band method. This ensures that the authentication does not rely on a shared secret but on the use of the public key and the corresponding private key. However, both the PSK and the PSPK methods have one important drawback. The keys are pre-configured, and therefore, the keys must be managed internally. An update may be required due to, e.g., the shared key being compromised. The management of these keys must be done manually or e.g. using a custom implementation on a local server. For this reason, a Trusted Third Party (TTP) is typically used. When using PSTTPPK, the TTP generates, for example, a certificate to link a public key to an entity. Then, other entities can, via the public key of the TTP, verify the certificate and thus the link between the public key and the entity. Thus, as long as the TTP is not compromised, entities can verify the authenticity of one another.

Securing a channel via encryption can typically be divided into two phases: a setup phase (session setup), and a runtime phase (session runtime). For example, the TLS protocol version 1.2 [87] uses a similar approach. During the session setup phase, a session encryption key is generated and both communicating parties are authenticated. This session key is, then, used by the symmetric-key algorithm to encrypt the communication between the two entities.

5.5.1 Session setup

The PSK method uses two random values padded to the PSK as input to a key derivation function (KDF) to generate a session encryption key. The PSK is not used as a session key such that the system cannot be compromised by attacking the session key. First, both the client and server generate and communicate to each other their random value. The random value is generated using the Hash-based Deterministic Random Bit Generator (Hash-DRBG). It is also a source of randomness recommended by NIST [8]. Then, a session encryption key is derived. The Hash-based Key Derivation Function (HKDF) is used as KDF.

While using public-key authentication, the session encryption key is often generated using the Diffie-Hellman (DH) technique. It is a public-key scheme that can create a shared secret over an unsecured channel. In the PSPK method, the Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) is used in this chapter. For this scheme, both entities generate a new set of public and private keys (ephemeral keys) and send each other the public key. Via the DH scheme, a shared secret is generated. To ensure the entity authentication of both the client and server, it is assumed that the ephemeral public keys are signed using the entity's respective public key. Both parties now have a shared secret, but, it is not used as the session encryption key for the same reason as in the PSK method. Thus, random numbers need to be generated and exchanged, and, a session key is derived using the key derivation function. The Hash-DRBG and HKDF algorithm are also used for this purpose in this method.

The PSTTPPK method is similar to the PSPK method with the exception that the public keys of the communicating entities are exchanged to each other and validated using the TTP's public key. Thus, during the setup phase, both entities send each other their certificate. This certificate is signed by the TTP. In this chapter, the use of X.509 certificates is assumed. Then, each entity validates the certificate by checking the signature. The remainder of the setup phase is the same as the setup of the PSPK method.

The granular approach to estimate the computation energy cost of the previously mentioned algorithms is presented in Table 5.3. Only the most computation-intensive operations are considered, similar to the approach of Saeed et al. [91]. Furthermore, specific optimisations to these algorithms, like Shamir's trick used in ECDSA, are not factored in. Next, the computation energy cost of the three session setup methods is presented in Table 5.4.

Table 5.3: The computation cost of the considered algorithms divided into basic operations (H: hash function on a Message (M), PMG: fixed-Point multiplication, PM: point multiplication, and PA: point addition).

Algorithm	**Computation cost**
Hash DRBG (R)	$4\|H(M)\|$
HKDF (KDF)	$8\|H(M)\|$
ECDSA - sign	$\|H(M)\| + \|R\| + \|PMG\|$
ECDSA - verify	$\|H(M)\| + \|PMG\| + \|PM\| + \|PA\|$
ECDHE	$\|R\| + \|PMG\| + \|PM\| + \|KDF\|$

Table 5.4: The computation cost of the considered session setup methods (R: Hash DRBG, KDF: HKDF, DHE: ECDHE, and DSA: ECDSA).

Session setup	**Computation cost**
PSK	|R| + |KDF|
PSPK	|R| + |DHE| + |DSA-sign| + |DSA-verify| + |KDF|
PSTTPPK	|R| + |DHE| + |DSA-sign| + 2|DSA-verify| + |KDF|

The communication energy cost is estimated for the three session setup methods and the results are presented in Table 5.5. Only the communication of basic objects is assumed. This is in contrast to the TLS protocol, where a lot of overhead messages are required to first agree upon a cipher suite etc.

Table 5.5: The communication cost of the considered session setup methods (C: certificate, R: random value, Y: elliptic curve point, and S: signature).

	Communication	
Session setup	**Input**	**Output**
PSK	$\|R\|$	$\|R\|$
PSPK	$\|R\| + \|Y\| + \|S\|$	$\|R\| + \|Y\| + \|S\|$
PSTTPPK	$\|R\| + \|C\| + \|Y\| + \|S\|$	$\|R\| + \|C\| + \|Y\| + \|S\|$

The total energy cost of the three session setup methods are calculated using the granular approach. First, the communication and computation energy cost are separately presented in Figure 5.5. Next, three combinations of the tested MCUs and wireless networks are made: nuc + LoRa (SF7), msp + Wi-Fi, and max + BLE (4.1). Only the best-case scenario for LoRa (SF7) is taken into account. The total energy consumption and the relative share of the computation and the communication are presented in Table 5.6.

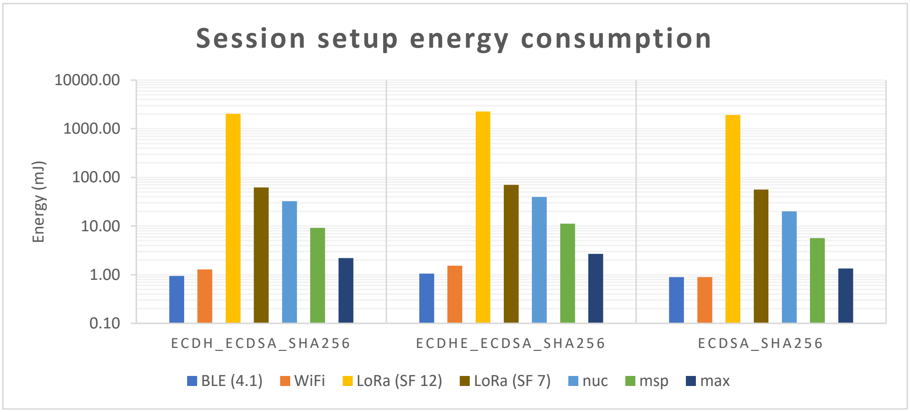


Figure 5.5: Energy consumption of the three configurations for the wireless networks and MCUs separately.

The PSK method requires the least amount of energy, and, the communication energy cost outweighs the computation cost for the nuc and max platforms, but not for the msp platform. The reason is that Wi-Fi has a higher energy efficiency than BLE and LoRa. For the PSPK and PSTTPPK method, the computation energy cost has a much larger impact than the communication energy cost. However, the LoRa network, which is more energy efficient, suffers more from sending e.g. the certificate. For the PSTTPPK method on the nuc platform, the energy cost is almost equally divided over the computation and communication energy cost. Note that the results are a rough estimation and they are probably lower than the actual energy consumption. We have only taken into account the best case scenario e.g. perfect wireless conditions, no header overhead, etc. However, these rough estimations are sufficient to show trends and compare different approaches.

Table 5.6: The total energy consumption and the relative share of the computation and communication energy cost for the three considered session setup methods and the three MCU and platform combinations.

Configuration	**PSK**	**PSPK**	**PSTTPPK**
nuc + LoRa (SF7)			
Total (μJ)	3119	62345	133845
Comp (%)	2.9%	75.6%	50.2%
Comm (%)	97.1%	24.4%	49.8%
msp + WiFi			
Total (μJ)	66	13257	19185
Comp (%)	74.8%	99.4%	98.1%
Comm (%)	25.2%	0.6%	1.9%
max + BLE (4.1)			
Total (μJ)	68	3414	5654
Comp (%)	22.3%	92.3%	79.6%
Comm (%)	77.7%	7.7%	20.4%

5.5.2 Session runtime

The AES algorithm is used to provide the encryption of data. In terms of computation cost, the AES block cipher modes are considered as basic operations. In terms of communication cost, only the AEAD block ciphers require additional data besides the message to be sent. Thus, an authentication tag of 12 B is taken into account. The total energy consumption and the impact of the computation and communication are presented in Table 5.7.

The communication requires significantly more energy than the computation. For example, the computation only takes about 2% to 6% of the total energy cost for the max platform. However, the total energy cost is almost spread evenly for the msp platform. The computation uses about 9% to 32% of the total energy cost for AES ECB, CTR and CBC, while it takes about 24% to

Table 5.7: The total energy consumption and the relative share of the computation and communication energy cost for the considered session runtime algorithms and MCU and platform combinations. (LoRa: SF7, BLE: v4.1)

Algorithm	**TX (AES)**				
	ECB	**CBC**	**CTR**	**GCM**	**CCM**
nuc+LoRa					
Total ($\mu J/B$)	67.66	67.67	67.68	67.78	67.80
Comp (%)	0.1%	0.1%	0.1%	0.3%	0.3%
Comm (%)	99.9%	99.9%	99.9%	99.7%	99.7%
msp+WiFi					
Total ($\mu J/B$)	0.15	0.15	0.16	0.25	0.24
Comp (%)	28.4%	31.0%	31.8%	57.3%	55.2%
Comm (%)	71.6%	69.0%	68.2%	42.7%	44.8%
max+BLE					
Total ($\mu J/B$)	0.81	0.81	0.81	0.84	0.84
Comp (%)	2.6%	2.7%	2.8%	6.0%	6.4%
Comm (%)	97.4%	97.3%	97.2%	94.0%	93.6%

Algorithm	**RX (AES)**				
	ECB	**CBC**	**CTR**	**GCM**	**CCM**
nuc+LoRa					
Total ($\mu J/B$)	27.11	27.12	27.13	27.23	27.25
Comp (%)	0.3%	0.3%	0.4%	0.7%	0.8%
Comm (%)	99.7%	99.7%	99.6%	99.3%	99.2%
msp+WiFi					
Total ($\mu J/B$)	0.45	0.46	0.46	0.56	0.54
Comp (%)	9.3%	10.4%	10.7%	25.7%	24.1%
Comm (%)	90.7%	89.6%	89.3%	74.3%	75.9%
max+BLE					
Total ($\mu J/B$)	0.87	0.88	0.88	0.90	0.91
Comp (%)	2.4%	2.5%	2.6%	5.5%	6.0%
Comm (%)	97.6%	97.5%	97.4%	94.5%	94.0%

57% for the AEAD ciphers. For the nuc platform that uses the LoRa network, the computation cost is almost negligible. It requires around 0.1% to 0.8% of the total energy cost. Note that the Wi-Fi wireless communication has the best theoretical energy efficiency of the considered wireless communication standards, as indicated in Table 5.2. Besides providing a rough and best case estimation, Wi-Fi has a much higher theoretical data throughput.

5.6 Energy Results for Digital Signatures

Another method of securing data is through the use of signatures. The following two algorithms are considered: the Schnorr signature scheme and the signcryption scheme. More details on the Schnorr and signcryption scheme can be found in Chapter 2. It is assumed that AES-CTR is used in the signcryption scheme.

The computation energy cost is again estimated using the granular approach. The required basic operations are listed in Table 5.8. In terms of communication, both schemes produce a signature of 64 B. Also, we assume that the public keys of both parties are pre-configured and do not require additional communication or verification.

Table 5.8: The computation cost of the considered public-key based algorithms divided into basic operations (H: Hash function on a Message (M), Y: elliptic curve point, PMG: Fixed-Point Multiplication, PM: Point Multiplication, PA: Point Addition, E: Encryption, and D: Decryption).

Algorithm	**Computation cost**
Schnorr - sign	$\|R\| + \|PMG\| + \|H(M+Y)\|$
Schnorr - verify	$\|PM\| + \|PMG\| + \|PA\| + \|H(M+Y)\|$
Signcryption - sign	$\|R\| + \|PMG\| + \|PM\| + \|H(M+3Y)\|$ $+\|E(M)\|$
Signcryption - verify	$2\|PM\| + \|PMG\| + \|PA\| + \|H(M+3Y)\|$ $+\|D(M)\|$

The energy efficiency in function of the data packet size is presented in Figure 5.6. It decreases with increasing message size, because, the public-key based operations only need to be performed once per message. As a reference, the energy efficiency of AES-GCM is added to the graph. In terms of efficiency, it requires a message size of about 5 kB to 50 kB to reach the efficiency of AES-GCM. However, we do not take the key establishment requirement of symmetric-key based end-to-end security into account. Furthermore, the coupon technique could be used to omit the required public-key based operations of both the Schnorr and signcryption scheme as explored by Winderickx et al. [120].

5.7 Derivation of the Minimal Session Period

Section 5.5.1 shows that the computation and communication energy impact of the session setup phase is considerable. However, the energy consumption

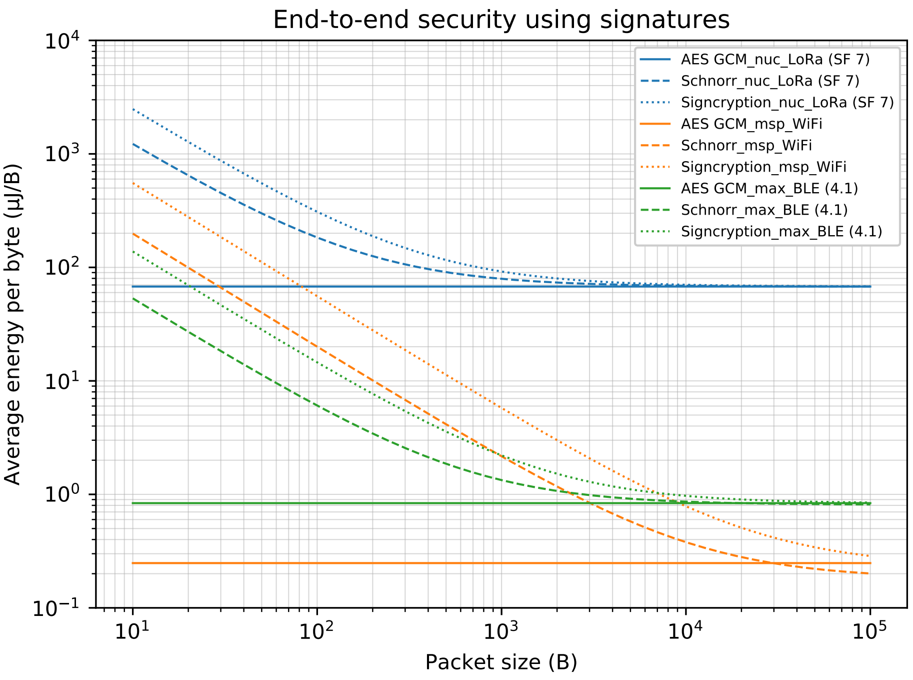


Figure 5.6: The energy efficiency of the Schnorr signature scheme and signcryption scheme as a function of the message size for each MCU and wireless network combination. The energy efficiency of AES-GCM is also added as a reference value.

impact can be minimised by looking at the big picture of providing a secured communication channel, i.e. taking into account that a session can be divided into a setup phase and a runtime phase. The impact can be reduced by controlling the frequency at which the session setup phase is executed. First, an equation for the minimum time period of a session is derived. The equation is then applied to two IoT applications: a wearable healthcare device and a weather station.

The total average power consumption P_{total} of a session is defined by the average power consumption of the session runtime P_{runtime} and the energy cost of the session setup E_{setup} divided by the time period of the session T, see Equation (5.3).

$$P_{\text{total}}(T) = \frac{E_{\text{setup}}}{T} + P_{\text{runtime}} \tag{5.3}$$

The average runtime power consumption (P_{runtime}) can be estimated using Equation (5.4). In a practical scenario, it can also be measured. In this equation, the transmission rate of the message, the total amount of application data in a message and the amount of additional data are denoted by, respectively, R_{app}, N_{app} and N_{add}. The data additionally generated by the encryption, e.g. the MAC, is counted as additional data. Note that it is assumed that the application only transmits data, and that the platform's and wireless network's idle energy consumption are negligible. In the equation, the amount of data is multiplied by the message transmission rate and the respective energy cost per byte for the communication (E_{TX}) and the computation (E_{AES}) aspect.

$$P_{\text{runtime}} = (N_{app} + N_{add})R_{app}E_{TX} + N_{app}R_{app}E_{AES} \tag{5.4}$$

The energy cost of the session setup is estimated using the results of Section 5.5. In a practical setting, a separate benchmark of the entire setup phase of e.g. the TLS protocol could be done to produce more accurate results.

As an example, we could state that the average power consumption impact of the session setup phase should be limited to 5% of the session runtime's average power consumption. This would then lead to Equation (5.5).

$$P_{\text{setup}} = 0.05P_{\text{runtime}} \tag{5.5}$$

Since $P_{\text{setup}} = E_{setup}/T$, the equation to calculate the minimal time period of the session can be derived, see Equation (5.6).

$$T_{\text{minimal}} = \frac{E_{\text{setup}}}{0.05P_{\text{runtime}}} \tag{5.6}$$

The total average power consumption as a function of the time period of a session is plotted in Figures 5.7 and 5.8 for the healthcare and the weather station scenario, respectively. The wearable healthcare device transmits 33000 kB of data every 10 seconds [119]. The weather station transmits about 224 B of data every half hour [89]. The following three configurations of the session setup and runtime phase are considered: PSK + AES-GCM, PSPK + AES-GCM, and PSTTPPK + AES-GCM. Additionally, the minimal time period based on the 5% rule that we introduced as an example is calculated for all configurations.

The healthcare scenario is plotted in Figure 5.7. The nuc and LoRa combination clearly does not fit this setting, because it takes too much energy to transmit the high amount of data. The best fit is the Wi-Fi network combined with the msp platform for all configurations. Furthermore, the msp platform has better results than the max platform, because, the Wi-Fi network chip features better energy efficiency. The minimal session period based on the 5% rule ranges from 10.1 s to 330.2 s. It is very low, because the healthcare scenario requires a lot of energy to send its application data. Furthermore, it should be noted that this time period is a lower bound.

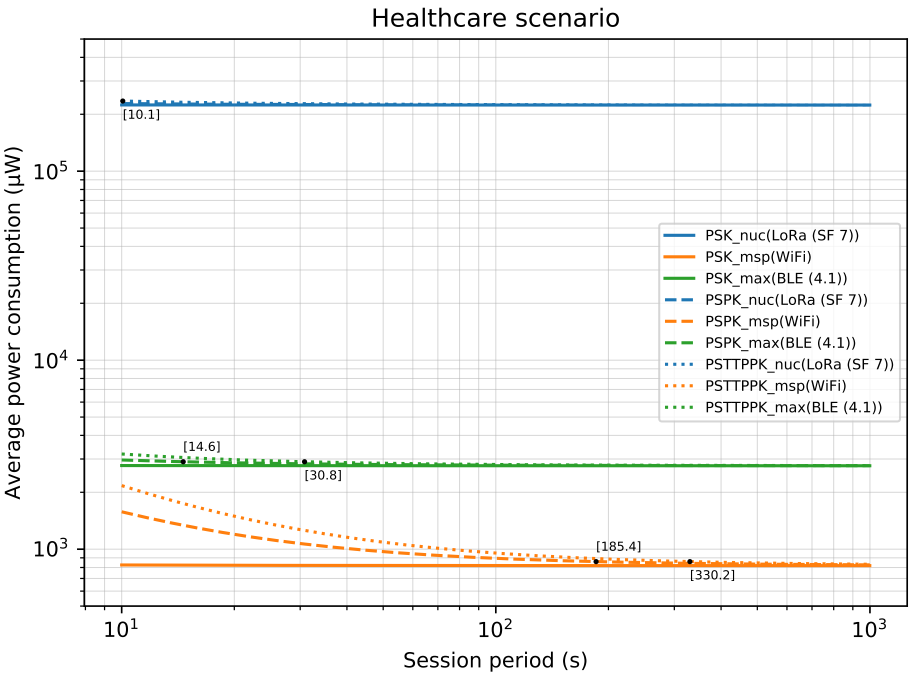


Figure 5.7: Average power consumption as a function of the session's time period for the healthcare scenario, where the minimal time period based on the 5% rule is plotted using a black dot.

The weather station scenario is shown in Figure 5.8. Depending on the position and range requirements of the application, all combinations are possible. The Wi-Fi network can provide the lowest average power consumption. The LoRa network consumes the most energy, but, it has the advantage of range over the BLE and Wi-Fi network. In this lower data throughput scenario, the minimal session period based on the 5% rule is considerably higher than in the healthcare scenario. It ranges from 1.9 hours to 99 days.

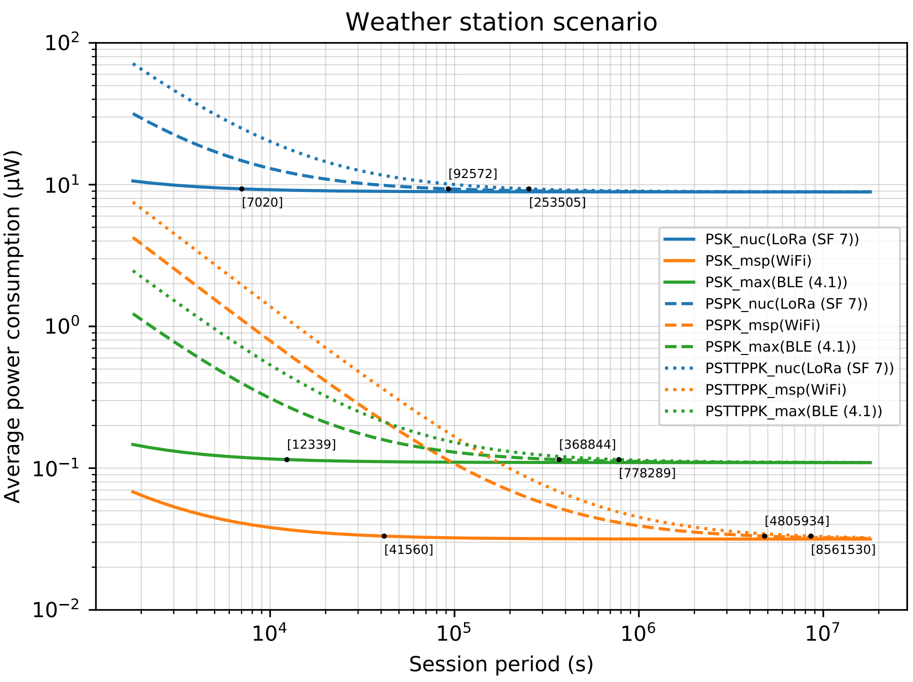


Figure 5.8: Average power consumption as a function of the session's time period for the weather station scenario, where the minimal session period based on the 5% rule is plotted using a black dot.

5.8 Conclusion

A granular approach was used to estimate the computation and communication energy cost of algorithms used to provide end-to-end security on the one hand and digital signatures on the other hand. Measurements were done on three platforms (nuc, msp and max) using three different wireless communication protocols (BLE, Wi-Fi and LoRaWAN). First, basic operations of the considered algorithms were identified and benchmarked. Then, the computation energy cost of these basic operations were evaluated. To explore the impact of both the computation and communication energy cost, two security techniques were explored: end-to-end encryption and digital signatures. For end-to-end encryption, both the session setup phase and the session runtime phase were taken into account. For the session setup, the energy results showed that the computation energy cost outweighed the communication cost in most evaluated scenarios. For the session runtime phase, this was the other way around. The results of the signature based security turned out to be not suitable for IoT applications without additional computation optimisation. The Schnorr and signcryption scheme reached the energy efficiency level of AES-GCM at about 5 kB to 50 kB of processed and transmitted data.

We also introduced an equation to calculate a recommended lower bound on the lifetime of a session to enable the developer to find a trade-off between the average power consumption and the side-channel resistance. In terms of security, the highest security level is achieved when the session is continuously renewed. On the other hand, the lowest average power consumption is obtained when the session is never renewed. In the equation, the developer can define the relative share of the application's total energy budget that may be used for renewing the security session. In our examples, we have chosen to dedicate 5% of the total energy budget to the security sessions. The equation was applied to the following two IoT scenarios: a healthcare and a weather station application. The relative energy share of the session setup phase in the overall energy consumption was almost negligible, i.e. lower than 5%, for minimal time periods in the range of 30 to 330 s for the different IoT platforms in the healthcare application. For the weather station application, the minimal time period for making the session setup energy negligible ranged from about 1.9 hours to 99 days for the different IoT platforms.

In summary, this chapter gave an overview of the energy impact of different security schemes for the IoT, taking into account both the computation and the communication energy. It also used this information to determine the minimal session period needed to make the session setup energy negligible. The chapter serves as a guideline for practitioners and researchers selecting the appropriate security algorithms and wireless communication protocols, and determining the minimal session period in order to minimise the overall energy consumption.

Chapter 6

Use cases on end-to-end security for wearable medical devices

The General Data Protection Regulation (GDPR) [78], strengthened by national law, enforces data protection and privacy for all individuals at the European level. As a consequence, sensor data measured on hospitalised patients should be protected against potential adversaries that retrieve or manipulate the data. Two methods on providing security are analysed in this chapter. First a proof-of-concept wearable medical system is secured via public-key cryptography in Section 6.1. In this section, proven network and security protocols are combined, efficiently implemented and evaluated. Secondly, a custom symmetric-key based security protocol is designed for a wearable medical system in Section 6.2. The same sensor node is used in the evaluation of both systems. The results of the two methods are compared in Section 6.3. Finally, a conclusion is given in Section 6.4.

6.1 End-to-end security via public-key cryptography

6.1.1 Introduction

The system under consideration in this section, consists of a wireless waterproof wearable device, communicating with a hospital server, developed in the wearIT4health project [82]. The monitoring system's features are determined based on (1) co-creation sessions with future users - healthcare professionals and potential patients, (2) market analysis and (3) the collaboration with three hospitals. The wearable device continuously monitors heart rate, blood pressure variation, breathing rate, oxygen saturation, skin temperature and human activity (intensity and posture). Raw data used for the estimation of these parameters are the electrocardiogram (ECG), the photoplethysmogram in three wavelengths (PPG), bio impedance (BioZ), the temperature (T) and the 3-axes accelerometers (ACC). These data are transmitted wirelessly from the wearable device on the patient, hereafter named patch, to the local hospital server. The patch is intended for use on nursing ward patients. It is designed to allow mobility for adult patients (>=18 years old) to provide physiological information. However, it is neither intended for use on critical care patients nor for diagnosis. The local server will process the measured data and use it for two applications: monitoring and reporting to the Electronic Medical Record (EMR). The EMR refers to the comprehensive medical records of an individual that are accessible in electronic form. Moreover, the local server can also notify healthcare professionals when physiological data fall outside specified ranges of selected parameters. The data measured by the wearable device are intended for use by healthcare professionals as an aid to monitor patients.

Our contributions can be summarised as follows:

- We implement end-to-end security in a wearable health monitoring system, intended for real-life use in a hospital, with a multitude of measured sensor data.
- We explore the impact on the energy consumption of different security schemes and implementations, as well as different wireless network protocols. As such, our work serves as a guideline for researchers and practitioners setting up a wearable medical sensor system or any battery-powered sensor system that needs to communicate a relatively large amount of data while providing end-to-end security.
- We present a proof-of-concept implementation of the resulting system and the corresponding measurement results.

First, related work is analysed in Section 6.1.2. Then, the system architecture is described in Section 6.1.3. For the selection of the most suitable wireless communication network for our system, low-power networks are compared in Section 6.1.4. Next, to ensure the protection of the measured data, the security requirements are analysed and used to create a security architecture, described in Section 6.1.5. Then, after the theoretical analysis in Sections 6.1.4 and 6.1.5, a practical exploration is presented in Section 6.1.6. It implements and compares the different security solutions using the most suitable wireless communication protocol. Finally, a conclusion is made in Section 6.1.7. My contribution to the work presented in this section is the study, the implementation and the evaluation of the considered protocols.

6.1.2 Related work

Two popular healthcare projects in research are CodeBlue [65] and MEDiSN [51], but many more projects and platforms have been devised, as analysed by Javdani et al. [49]. CodeBlue [65], proposed by Malan et al. in 2004, is an ad hoc sensor network infrastructure for emergency medical care. The authors used MICA2 motes to monitor the pulse oximetry of patients. MEDiSN [51], created by Ko et al. in 2010, provides medical emergency detection in sensor networks. The authors used the Sentilla Tmote Mini platform for two use cases: ECG monitoring, and pulse oximetry with LCD screen. However, in more recent works, researchers also began looking into optimisations. For example, Samie et al. [94] used data compression in a wearable ECG monitoring platform to reduce the overall energy consumption and it resulted in a device that could theoretically monitor the ECG of a patient for 10 days using a 400 mAh battery.

As stated, the number of healthcare platforms in research is increasing, as interest in integrating portable devices continues to grow. Nevertheless, the security issue is still a concern. In this work, we analyse the implementation and integration of a wearable healthcare platform in a hospital while providing true end-to-end security to protect the patient's personal data.

End-to-end security is important to take into account when dealing with sensitive data. Without end-to-end security, intermediate devices like gateways can compromise the integrity and/or the confidentiality of the transmitted data. This was already specified by the earliest works about patient monitoring. For example, both the CodeBlue [65] and MEDiSN [51] projects expressed the importance of security by referring to privacy issues and national regulations like the 1996 Health Insurance Portability and Accountability Act (HIPAA), but neither provided end-to-end security. In the first published paper of the CodeBlue [65] project, security was put as future work and it was revisited by

Kambourakis et al.[50] in 2007. It was, however, to our knowledge, never fully implemented with end-to-end security. In the MEDiSN [51] project, the authors did implement secured communication between the entities in the network but they did not provide end-to-end security.

Security in constrained environments has been a hot topic for many years. For example, the concept of a delegation server or trusted third party to offload the authentication and authorisation computations from constrained devices has been researched extensively by, e.g., Hummen et al. [46], Raza et al. [85], and Kerberos [73]. The main motivation for using a delegation server is that the impact of the authentication and authorisation protocols on the energy consumption of constrained devices is too large. Furthermore, it is often assumed that the constrained device has a preconfigured secure communication channel with the delegation server. The delegation server can setup connections with remote parties and it can then pass along the required session information to the constrained device. After this procedure, the constrained device and remote server can securely communicate. By offloading the authentication and authorisation, the constrained device does no longer need to utilise the computationally expensive algorithms that are typically used for these purposes like the asymmetric-key cryptography of RSA or Elliptic Curve Cryptography (ECC). For example, the paper by Moosavi et al. [70] showed that it is also possible to use the fog to provide a distributed set of delegation servers to offload the authentication/authorisation. The advantage is that a Denial-of-Serice (DoS) attack is less likely because the delegation server is no longer centralised.

The issue with the delegation server is that it is a target for attackers to compromise one or more constrained devices. In this work, we avoid the use of a delegation server to narrow the range of targets available to the attacker. We analyse and limit the impact on the constrained device by leveraging the newest technologies and techniques with regards to communication security. This work extends the work-in-progress results presented in [119].

6.1.3 System model

The system architecture, presented in Figure 6.1, consist of three entities: the patch, the local server and the hospital infrastructure. In this section, first, the patch is described. Then, the functionalities of the local server are given. Finally, the workflow of the system, that describes how the patch is used, is given in Appendix A.

The local server also interconnects with the hospital infrastructure for monitoring, managing and reporting purposes. Clinical staff can access the

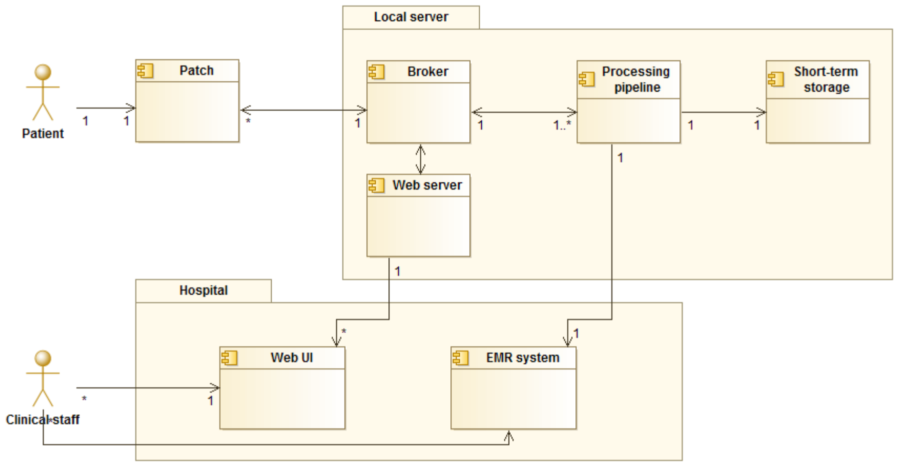


Figure 6.1: Overall system architecture (1: one entity; *: multiple entities).

patient's data through a web-based user interface (Web UI) or through the EMR system. We do not elaborate further on the hospital infrastructure because it depends on specific choices made by the IT department of the hospital. Nevertheless, the system described in this section is validated on three different hospital networks and can therefore be considered to be broadly usable.

Patch

The patch is composed of three different parts: (1) two textile electrodes, attached to the housing via snap buttons, (2) an electronic board and (3) a battery. Both the electronic board and the battery are contained in the housing. The electrodes have to be changed for each patient as they could lead to cross contamination if used on multiple patients. The electronic board contains a microcontroller, a network interface and all the required on-board sensors, namely ECG, PPG, BioZ, T and ACC, as introduced in Section 6.1.1. The battery is easily expendable when the patient is wearing the patch in order to facilitate its continuous use.

The on-board sensors produce about 3.3 kB of raw data every second. In order to reduce the data overhead and to lower the number of times the device should awake from sleep (wake-up overhead), 33 kB of data is sent to the server every 10 seconds. The raw data is not pre-processed on the patch, because, the availability of the raw data was requested by the wearIT4health project for the development of the algorithms on the Local server.

Local Server

The local server runs three applications: a message broker, a processing pipeline and a web server. The message broker is used by the patch to publish its raw data, by the processing pipeline to convert the raw data into vital parameters, and by the web server to display these vital parameters. Furthermore, commands to and from the patch and the web server are also communicated via the message broker. Regarding the conversion of raw data to vital parameters, a 30-second time window is used. At the start of each window, a copy of the raw data is saved into short-term storage. It can later be used for analysis. Then, within each 30 seconds, raw data is first preprocessed to eliminate noise and then processed to extract the vital parameters of the patient. Furthermore, the processing pipeline is responsible for assessing the patient's state. It estimates the early warning score (EWS) for each patient and triggers notifications to the clinical staff if abnormal conditions are detected in the patient's vital parameters. The EWS is estimated every 10 s. Depending on the instructions of the clinical staff, the pipeline can also update the patient's electronic medical record (EMR) with specific data points.

6.1.4 Wireless communication trade-offs

The requirements of the patch can be summarised as follows. It should be wirelessly connected to the local server and comfortable for the patient to wear. That means that it is necessary to reduce the size and the weight of the patch, which is directly related to the battery size. Therefore, the way of processing and transmitting data should be power efficient. It should be possible to install a patch on each patient in the hospital, thus, the network should be designed to handle a large number of patches. As patients can move inside the hospital, the network should cover the entire hospital. This sections makes a theoretical comparison of different network topologies and wireless communication protocols in order to select the most suitable solution for our application scenario.

Network topology

There are three main types of network topologies:

- **Ad-hoc**: one device can communicate directly with every other device within its range. This topology is adapted to a relatively small number of devices. As the link is direct between two devices, the range is limited by the protocol and the power of the signal transmission. The advantage is that the implementation of this type of network is relatively simple as it does not require protocols for e.g. routing.

- **Mesh**: the mesh topology is an extension of the ad-hoc topology. Some devices can act as router and relay data between two nodes. These routers extend the coverage at the expense of an increased power consumption for transferring data. Furthermore, all devices share the same network, thus, they can easily be integrated into the network.
- **Star**: a central access point communicates directly with all peripherals in its range. There can be more than one access point linked together to extend the coverage to the entire area. Generally, theses access points are not power constrained because they are plugged into the electrical grid. This kind of network can be seen as a mesh with only access points as routing nodes. Thus, the peripherals in the network use less power than in the mesh network, but, the network requires an extensive network infrastructure to cover the entire area.

In the scenario that we consider in this section, there is only one local server collecting all the data. Moreover, taking into account the typical size of a hospital, an ad-hoc network topology cannot ensure connectivity for all patch devices. The mesh topology is also less suitable because of its increased power consumption and the fact that the coverage relies on the number of devices and their distribution. Therefore, the star topology is the most convenient for our application at the expense of the cost of fixed access points in the hospital.

Protocols

There are a lot of existing wireless communication protocols, e.g. Bluetooth Low Energy (BLE), Wi-Fi, ZigBee, LoRaWAN and Sigfox. These protocols can be divided into different types of networks as explained in Chapter 2: Wireless Personal Area Networks (WPAN), Wireless Local Area Networks (WLAN) and Low-Power Wide-Area Networks (LPWAN). An overview of the network protocol specifications is given in Table 6.1.

The WPAN protocols like BLE and ZigBee have low throughput (250 kbps - 1 Mbps) and operate over short ranges indoor (10 - 100 m), but, they have the advantage to be low-power. In contrast, a WLAN protocol like Wi-Fi is well adapted to send large amounts of data (54 - 150 Mbps) over slightly larger distances (100 - 250 m). LoRaWAN and Sigfox can transmit data over very long ranges of a few kilometres. However, they are not optimised for indoor use and feature typically very low throughput (300 bps - 50 kbps) and can handle only a limited amount of data. Furthermore, some LPWAN networks like Sigfox are hosted by an operator which results in a paid subscription.

LoRaWAN and SigFox are clearly not applicable for the application studied in the chapter at hand. We selected the other three protocols, Bluetooth, ZigBee and Wi-Fi, and made a theoretical comparison through a set of the most recent chips available on the market: BL652-SA, RN4020, BT800 and LM931 as Bluetooth chips, JN5168, MGM11, Xbee ZB SMT and EM351 as ZigBee chips, and LM821-0463, WGM160, CX53111 and CC3120 as Wi-Fi chips.

Table 6.1: Comparison of wireless communication protocols suitable for indoor application.

Features	**BLE**	**ZigBee**	**Wi-Fi**
IEEE specification	802.15.1	802.15.4	802.11b/g/n
Frequency band	2.4 GHz	868/915 MHz 2.4 GHz	2.4 GHz
Data rate (Mbps)	1-3	0.250	54-150
Nominal range (m)	10-100	10-75	100-250
Max active devices	8	65000	2007
Security	AES-CCM	AES-CCM	WPA2
Power consumption			
TX (µW/kb)	3.7-88.8	108.2-435.6	3.8-465
RX (µW/kb)	3.6-72.6	129.3-369.6	3.1-120.8
Idle current (µA)	2-200	0.7-2.5	690
Max payload (B)	339	102	1500
Max overhead (B)	158/8	31	58
Efficiency	94.46%	76.67%	96.28%

We made an estimate of the efficiency of the 3 protocols to send the amount of data required by our system architecture (33 kbytes). The efficiency is defined by the payload size ($Ndata$), the number of packets ($Npackets$) and the data overhead per packet ($Noverhead$) [59], see Equation (6.1). The number of packets is defined by the fragmentation of our data that is required in the respective wireless communication protocol. Then, for each packet sent, the header overhead needs to be added.

$$\text{Efficiency} = \frac{Ndata}{Ndata + (Npackets * Noverhead)} \tag{6.1}$$

As can be seen in the last line of Table 6.1, it turns out that ZigBee is less efficient in our scenario, because, we need to send a large amount of data.

It also takes more time to transmit our data via the ZigBee protocol, which results in less time in low-power sleep mode and consequently in more energy consumption. Regarding the network size, ZigBee is the one that can deal with the largest number of nodes. However, bandwidth needs to be shared between all patches which greatly reduces the number of nodes ZigBee can handle, since each patch requires a high data rate.

On the other hand, the average power consumption per byte results of the BLE and the Wi-Fi network overlap, but, Wi-Fi has a better coverage and can deal with a larger number of devices per cell. Additionally, Wi-Fi has three advantages over BLE for our setup: (1) less gateways need to be deployed because the range is larger, (2) a Wi-Fi infrastructure is often already present in hospitals, and (3) the patch could also easily be used for monitoring patients at home, as Wi-Fi is already widespread in almost every house.

It should be mentioned that these protocols could be extended to more complex structures that can improve the number of devices that can connect to the network at the expense of more access points. Nevertheless, we conclude this section by selecting Wi-Fi as the most suitable protocol for our application, given that 33 kbytes of data need to be sent every 10 seconds, targeting a low energy consumption and the ability to handle many patches.

6.1.5 Security trade-offs

The objective is to secure the entities and data flows as presented in Figure 6.1. This section concentrates on two entities, the patch and the local server, and the data flow between them. First, the security requirements are compiled using the STRIDE threat modelling technique. Next, three techniques are evaluated to provide an end-to-end secured communication channel. Finally, the online and offline security requirements in terms of access control are discussed. The theoretical analysis made in this section, will be brought to practice in Section 6.1.6, which implements and compares the different security solutions.

Security requirements

At the start of developing the security architecture, the relevant threats to the system must be identified. We use the STRIDE threat model [101] and the STRIDE-per-interaction technique to find the threats. An abbreviated compilation of the threats is given below, according to the six categories: Spoofing, Tampering, Repudiation, Information disclosure, Denial-of-service and Elevation of privilege:

- **Spoofing** refers to a scenario in which data are disguised as coming from a known, trusted source, while they are coming from an unknown, untrusted source. In our setting, the system can contain multiple patches but only one local server. The patch measures sensitive data, which is communicated to and analysed by the local server. In order to prevent spoofing, mutual authentication is required between the patch and the local server. This can be achieved through *entity authentication* and *data origin authentication* in order to validate the entities and the dataflow, respectively.

- **Tampering** is the act of altering or damaging data while being communicated from one entity to the other. In our system, the data collected by the local server is used for operations like monitoring and assessing the health of a patient. It is, therefore, important to prevent tampering by ensuring the *data integrity* of the communication flow.

- **Repudiation** is the rejection or the refusal of acknowledgement of a previously made agreement. Preventing repudiation typically involves the use of logging to prevent interactions to be denied. In our scenario, repudiation protection can be enabled on the server by using logs of interactions. It is, however, difficult to implement on the patch, since the patch is a constrained device in which the storage is already filled with application code and measured data. Nevertheless, the local server is the central device that is involved in all interactions, which means that logging on the local server prevents the repudiation of all interactions.

- **Information disclosure** refers to an attacker gaining unauthorised access to valuable information. Since the measured vital sign parameters are personal in the considered patient monitoring system, *data confidentiality* is imposed by law [78]. Furthermore, the impact of the attack should be limited if a device is compromised by e.g. side-channel attacks. A compromised device should not affect previous or future patients. This can be achieved by providing Perfect Forward Secrecy (PFS). PFS ensures that the previous sessions are not compromised when the current session is compromised. Confidentiality protection is, therefore, required for the data transmitted and stored by the patch as well as the local server. For example, 1 MB of storage could be used to facilitate about five minutes of data (30 kbytes every 10 seconds).

- **Denial-of-service** attacks intend to make a network device unavailable to its intended users by flooding it with traffic. The considered monitoring system operates in the internal secured network of the hospitals where Denial-of-service attacks are highly unlikely. However, disconnections or weak reception can render the patch unable to send its data. It is, therefore, advised to reserve storage on the patch to store data temporarily.

- **Elevation of privilege** concerns the situation in which a lower-privilege user accesses functionality or content reserved for higher-privilege users. In order to protect the system from this threat, *authorisation* should be implemented. This is important because only authorised devices should be able to upload data for specific patients, and only authorised personnel should have access to the data of specific patients.

In summary, based on the STRIDE approach, the five desired cryptographic properties that are identified in our system are *entity authentication*, *data origin authentication*, *confidentiality*, *data integrity* and *authorisation*.

End-to-end secured communication

In most end-to-end secured tunnels, two different types of protocols are used: key establishment and secure communication. Key establishment is typically a handshake which is based on either symmetric-key cryptography, public-key cryptography or both. It can provide the required entity authentication. During the handshake, the authenticity of both communicating identities is confirmed while a session key is derived. The session key can then be used to secure the actual communication. This is typically done using a symmetric-key encryption algorithm like AES [26]. AES is a block cipher which can be used in different modes of operation to achieve a specific set of security requirements. For example, CTR (Counter) mode and CBC (Cipher Block Chaining) mode provide confidentiality, and GCM (Galois/Counter Mode) and CCM (Counter with CBC-MAC Mode) are authenticated modes of encryption that ensure both confidentiality and authentication. In the following paragraphs, we discuss three methods for key establishment that enable end-to-end security. In our proof-of-concept implementation in Section 6.1.6, we compare different cipher suites based on these three approaches.

Symmetric-key approach The Pre-Shared Key (PSK) technique is often used in constrained environments for entity authentication and key establishment, because, it is very efficient. PSK is based on symmetric-key cryptography, where both communicating entities share a key, and this shared key is used to generate a master key. The key is shared through an out-of-band communication channel, e.g. through configuration in advance. The session keys that are used to encrypt the data are derived from the master key. The authentication is based on the implicit use of this master key. If an entity cannot derive a session key, it is assumed that it does not know the master key, and thus, the entity is unable to authenticate itself. In order to prevent attackers from performing a brute-force attack, it is important to regularly update the key and to use key lengths that are considered to be long enough for the envisioned protection level [74].

Public-key approach A technique based on public-key cryptography that is often used during the handshake, is the combination of the Diffie-Hellman (DH) protocol [29] for key exchange and the RSA [88] algorithm or the Elliptic Curve Digital Signature Algorithm (ECDSA) for entity authentication. Public-key cryptography uses two separate keys for each entity: a private and a public key. For entity authentication, the private key is used to sign data and the public key is used to verify the signature. If the verification succeeds, the data are authenticated to be originating from the corresponding entity. Moreover, the public key is often packed into a certificate which contains the public key, the corresponding entity's information, and a signature. This signature is generated by a trusted third party such that it can be validated by all entities.

Combined approach In a more recently proposed technique, a combination of PSK and Diffie-Hellman is devised [114]. It uses DH key exchange authenticated with a PSK. This way, the authentication is more efficient than the approach based on public-key cryptography only. By using DH, perfect forward secrecy can be achieved.

Access control

Access control is important for the offline and online security of entities. We denote offline security as the secure commissioning of the devices, and online security as access authorisation.

Local server The IT department of the hospitals will be responsible for hosting and managing the local server. It can be continuously monitored and no unauthorised physical and network access should be possible. At minimum, authorisation measures like login credentials should be used to limit access to the local server.

Patch Access to the patch cannot be controlled by the IT department since it is used on patients. Attackers, thus, may have physical access. Steps should be taken to secure all available interfaces of the patch, e.g. using authenticated firmware and password protection.

6.1.6 Proof-of-concept implementation + measurement results

The protocols that are used in the proof-of-concept implementation are presented and mapped to the OSI model in Figure 6.2. First, the Wi-Fi network was chosen for its efficiency and to facilitate interoperability and integration in hospitals. Secondly, we opted for using TCP to enable reliable communication over the IP network. Next, MQTT was used to provide a lightweight and proven messaging transport. It is also a standardised protocol of the International Organisation for Standardization (ISO) [107]. Finally, the TLS protocol is used to secure the MQTT communication. It is a protocol designed by the Internet Engineering Task Force (IETF) [87].

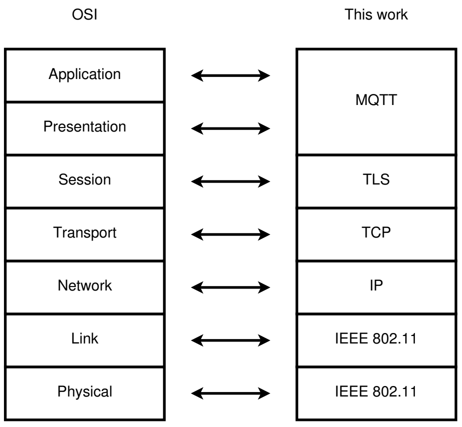


Figure 6.2: Selected protocols for the communication channel.

The patch is implemented using a custom platform. This platform features the MSP432P4011 [109] as application MCU, CC3120 [108] as network interface, and the sensor ICs required to measure the vital parameters described earlier. The MSP432P4011 is a SimpleLink Ultra-Low-Power 32-Bit Arm Cortex-M4F MCU with Precision ADC, 2MB Flash and 256KB RAM. The CC3120 is a SimpleLink Wi-Fi Network Processor. It features an ARM Cortex-M3 MCU that can completely offload the Wi-Fi and Internet Protocols from the application MCU.

The local server is Linux based. The software on the server uses the container technology of Docker [30]. The server runs RabbitMQ [83] as message broker and TimescaleDB as database. This combination has been validated to cope with the amount of data generated by 500 monitored patients using a single machine with 8GB of RAM and 4 CPU cores. The WebServer is implemented in

Java using the Spring framework. The WebServer serves an Angular application to manage, monitor and inspect near real-time and historical data. Finally, the streaming engine is a combination of Java code for the aggregation and routing logic, and MATLAB/C++ code for the actual signal processing algorithms.

Security overhead

To achieve the five required cryptographic properties, identified in Section 6.1.5, we selected four cipher suites for our analysis:

1. TLS_PSK_WITH_AES_128_GCM_SHA256
2. TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256
3. TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
4. TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

For secure communication, all cipher suites use AES, either in GCM or in CBC mode, in combination with the SHA256 hash function. For key establishment, three different cipher suite approaches are used as described in Section 6.1.5. The first cipher suite uses only symmetric-key cryptography based on the PSK approach. It does not provide Perfect Forward Secrecy (PFS); it is primarily added for reference. The second cipher suite uses a combination of symmetric-key and public-key cryptography for the key handshake. It is based on Elliptic Curve Diffie-Hellman Exchange (ECDHE) in combination with PSK. The third and fourth cipher suite correspond to the public-key approach. The third cipher suite uses ECDHE in combination with RSA, while the fourth cipher suite uses Diffie-Hellman Exchange (DHE) in combination with RSA. The second, third and fourth cipher suite provide all the required properties of an end-to-end secured channel including Perfect Forward Secrecy (PFS). The authorisation aspect is implemented by using user/password login credentials in the MQTT protocol. After successfully establishing the secured channel, the patch needs to provide these credentials to gain access to the MQTT broker.

The selected key sizes are a 2048-bit RSA key pair, a 256-bit ECC key pair, a 128-bit session key and a 256-bit hash.

The TLS protocol is implemented in two ways. For the first and second cipher suite, we have used the mbed TLS library (v2.11.0) [4]. Moreover, the SECP256K1 curve was used for the ECDHE implementation of this library. The third and fourth cipher suite, using public-key cryptography for both the authentication and key establishment, are hard to implement on the MCU because of storage and memory constraints. For this reason, we opted for the secure socket feature of our CC3120 network processor. It provides a public-key based TLS protocol implementation. In this scenario, the TLS protocol is offloaded from the MCU to the network processor. Moreover, the network processor uses a hardware accelerator for the RSA and the AES algorithm. The ECC curve used by the CC3120 chip is SECP256R1. Analysing the side-channel resistance of the mbed TLS library and the network processor is not in the scope of this chapter, though, they feature some countermeasures according to the available documentation.

The performance of the key handshake depends on the used algorithms and hardware, presented in Figure 6.3. The first cipher suite based on the PSK approach takes the least amount of time, about 60 ms, to establish a session. The other cipher suites involve public-key algorithms that are more computationally expensive. The MCU is a low-power processor that is limited in performance. This is notable in the execution of the ECDHE_PSK handshake. It takes about 2.6 seconds to establish a session. The secure socket feature of the network processor is much faster than the MCU for these types of calculations since it can perform the handshake under 0.5 seconds for ECDHE_RSA and DHE_RSA. The elliptic curve discrete logarithm based version (ECDHE_RSA) is about twice as fast as the discrete logarithm based version (DHE_RSA). This may be because of the available accelerators. However, the exact details of the implementation of the network processor is not known.

The estimated energy results of the handshake process are shown in Figure 6.4. The energy results where generated using the performance results and the specifications of the MCU and the network processor. The first cipher suite (PSK) is the most lightweight solution, using about 1.5 mJ. The second cipher suite (ECDHE_PSK), which is the slowest of the four cipher suites, does not consume the most energy. It uses the low-power MCU for its calculations, resulting in only 53 mJ. The third cipher suite (ECDHE_RSA), executed on the network processor, is much faster, but consumes much more energy to do the calculations. It takes around 147 mJ to perform a handshake. The fourth cipher suite (DHE_RSA) is both fast and energy efficient in comparison to the other public-key based handshakes. It uses about 32 mJ to establish a connection.

The implementation overhead of the TLS protocol is given in Table 6.2. Results were derived from the Memory Allocation report provided by Code Composer Studio. The first and the second cipher suite require additional code on the MCU because of the mbed TLS library. The first cipher suite (PSK) requires about 40 kB of storage and 5 kB of memory (stack and heap). The second cipher suite (ECDHE_PSK) uses an additional amount of around 17 kB of storage and 1 kB of memory in comparison to the PSK cipher suite. If the TLS

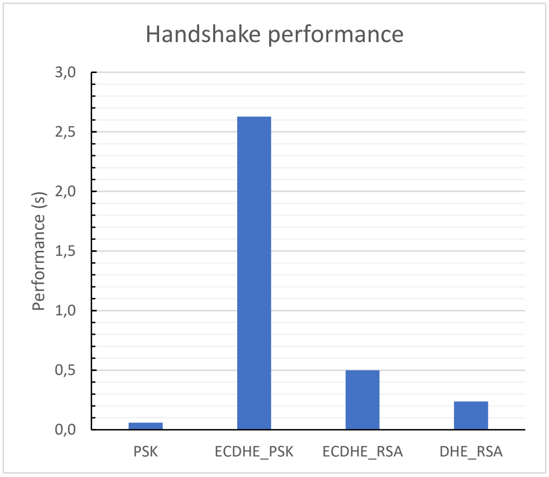


Figure 6.3: Performance evaluation for each of the handshake configurations, where PSK, ECDHE_PSK, ECDHE_RSA and DHE_RSA stand for the handshake protocols in the first, the second, the third and the fourth cipher suite introduced in this section. The first and the second approach are implemented on the MCU, while the third and the fourth approach are implemented using the secure socket feature of the network processor.

protocol is offloaded to the network processor, the MCU only requires about 1 kB of additional storage. This is the case for the third and the fourth cipher suite (ECDHE_RSA and DHE_RSA).

Table 6.2: Implementation overhead required for security protocols on the MCU.

Configuration	**Storage (kB)**	**Stack (kB)**	**Heap (kB)**
PSK	39.6	1.0	4.1
ECDHE_PSK	56.6	1.1	5.0
(EC)DHE_RSA	1.1	0.0	0.0

Secure communication

In terms of implementation size, the firmware code related to the application uses about 41 kB of storage and 142 kB of memory. In total of 2% of Flash and 55% of RAM memory of the MCU is used. This leaves enough room for the other TLS implementation configurations.

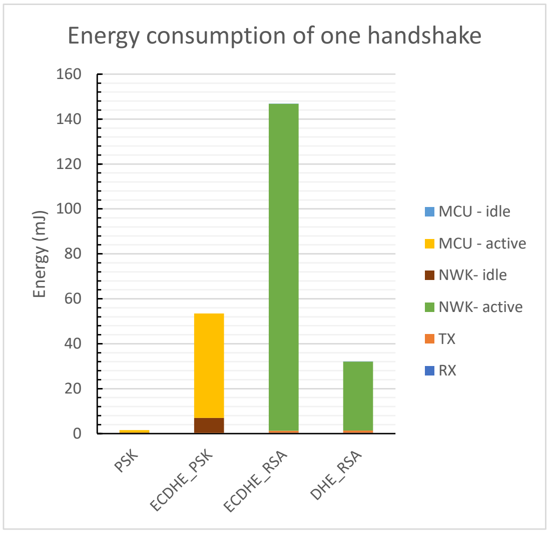


Figure 6.4: Estimated energy requirements for the each of the handshake configurations, where MCU, NWK, TX and RX stand for the energy consumption of the MCU, the energy consumption of the network processor and the energy consumption for transmitting and receiving data, respectively. A distinction is made between the energy consumption when the component is idle and when the component is active.

On powering up the patch, it first establishes a secured connection with the server. This connection is maintained for as long as the patch is used. The handshake process of the TLS protocol is, therefore, only executed once. Next, the patch requests its session parameters (e.g. the UID). Finally, the patch starts measuring and periodically sends data to the local server. Every 10 seconds, the sensor data is compiled and pushed to the server using MQTT. For the remaining time, the platform is measuring the vitals and listening for commands from the local server.

The average power consumption and lifetime estimation of the patch is measured and compared in Table 6.3 to related work introduced in Section 6.1.2. For our work, we use the results of the fourth cipher suite (DHE_RSA). The lifetime estimates are based on a 400 mAh battery. The results of related work were compiled and estimated using the electrical parameters available in those papers. Our result was measured using the Keithley 2000 Ammeter. Our platform consumes about 20% to 33% less than the other platforms that measure only the pulse oximetry (PO) parameter (CodeBlue and MEDiSN PO). However, it consumes about 6 to 12 times more power than the platforms which measure only the ECG parameter (MEDiSN ECG and the platform of Samie et al.). Nevertheless, it is difficult to compare our platform to related work, because, we have the disadvantage of measuring a larger range of parameters, and the advantage of using newer and more energy-efficient hardware.

The energy required for the key handshake using the fourth cipher suite (DHE_RSA) is only around 5% of the energy required for measuring and reporting the vital parameters of one period. In one period where the patch measures and reports the data, the patch uses about 698.2 mJ. The DHE_RSA based handshake process requires only 32 mJ. Given that the handshake only needs to be performed once in the lifetime of the patch, we can conclude that the energy consumption for providing end-to-end security is negligible to the energy spent during the entire lifetime of the patch.

Table 6.3: Comparison of average power and lifetime estimation of the platforms referred to in Section 6.1.2, with the following vital parameters: the pulse oximetry (PO), the electrocardiogram (ECG), the photoplethysmogram in three wavelengths (PPG), the bio impedance (BioZ), the 3-axes accelerometers (ACC) and the temperature (T).

Project	**Vitals**	**Power (mW)**	**Lifetime (days)**	**End-to-end security**
CodeBlue [65]				
	PO	87.78	0.63	no
MEDiSN [51]				
	ECG	11.29	4.87	no
	PO	105.30	0.52	no
Samie et al. [94]				
	ECG	5.87	9.36	no
This work				
	ECG, PPG, BioZ, ACC, T	69.82	0.79	yes

6.1.7 Conclusion

A wearable health monitoring system intended for real-life use in a hospital with a multitude of measured sensor data is designed and implemented. Using this system, six vital parameters are continuously being monitored: hearth rate, blood pressure variation, breathing rate, oxygen saturation, skin temperature and human activity (intensity and posture). Furthermore, an early warning score is added to notify clinical staff if abnormal conditions are detected. Additionally, clinical staff are able to use the system to report specific vitals to the Electronic Medical Record. The system is validated using a proof-of-concept implementation tested in three different hospitals. In comparison to related work, our patch provides a more energy-efficient monitoring system, taking into account that it supports a significantly higher amount of sensor data.

The system adds two custom entities to the hospital: the patch and the local server. The patch is a wireless wearable battery-powered device, and the local server is a Linux based device hosted by the hospital. The patch contains a MSP432P4011 MCU, a CC3120 network processor and the required on-board sensors. The impact of the wireless network protocol and the security architecture on the energy consumption is explored, resulting in the choice for Wi-Fi as the most efficient wireless communication protocol for our use case. Furthermore, to provide the end-to-end security between the patch and local server, adhering to the security requirements determined through the STRIDE threat modelling approach, the TLS protocol is chosen. Four cipher suites of the TLS protocol are analysed via two implementations: the mbed TLS library on the MCU and the secure socket feature on the network processor. Offloading the security to the network processor shows to be the most optimal solution. Furthermore, by leveraging long session lifetimes, the energy consumption overhead of establishing a true end-to-end secured channel is deemed negligible.

6.2 End-to-end security via symmetric-key cryptography

6.2.1 Introduction

In a practical setting, a device carried by the patient, e.g. a smartphone, can be used to collect data wirelessly from the wearable sensors. In the first place, it is important to guarantee end-to-end security between the wearable sensor device and the patient's device. In addition, wirelessly associating a wearable sensor device to a patient's device, which is typically done by hospital staff, should not violate the anonymity of the patient. With these security requirements in mind, this section proposes a symmetric-key security protocol. Besides the focus on end-to-end security and anonymous association, the protocol also concentrates on the minimisation of the communication bandwidth and the required computation power of the wearable device. The section explains the different protocol steps for both anonymous association and end-to-end secured communication. Further, a thorough security analysis and evaluation of the required communication and computation cost are performed. A comparison to previously proposed solutions is shown to be favourable for our protocol.

The outline of the section is as follows. Section 6.2.2 gives an overview of related work in the field of security protocols for healthcare. Section 6.2.3 describes the envisioned system and the security requirements. The different steps in our proposed solution are elaborated in Section 6.2.4. An evaluation of the security on the one hand, and the computation and communication requirements of the protocol on the other hand, are given in Sections 6.2.5 and 6.2.6. Finally, Section 6.2.7 concludes the section and points out to future work.

My contribution to the work presented in this chapter was the refinement of the protocol, after it was originally proposed by Prof. An Braeken. Further, I was responsible for the implementation and evaluation of the protocol.

6.2.2 Related work

A lot of key agreement schemes for healthcare systems or similar applications have been described in literature. In this overview of related work, we focus on the most recent schemes and explain the differences with respect to our proposed scheme.

In a recent study of Kumar et al. [57], a symmetric-key based key agreement scheme between smart sensor nodes and a home gateway in a Home Area Network (HAN) is proposed. The scheme is designed to provide anonymity, unlinkability, mutual authentication, integrity, perfect forward secrecy, and resistance to general attacks like replay attacks, impersonation attacks and man-in-the-middle attacks. Furthermore, the authors argue that a key agreement scheme in a HAN network should be automatic without human intervention. This leaves the gateway the most interesting attack target as it sits at the centre of the communication model of a HAN. In our setting, the patient is hospitalised and accesses the sensor nodes via a smartphone. The data are highly sensitive and should, therefore, only be accessible via human intervention, e.g. via an authentication step.

The communication model of Shuai et al. [102] is similar to ours as they provide a key agreement scheme for remote patient monitoring. In this scheme, the sensor nodes are connected in a Wireless Body Area Network (WBAN) to a gateway which translates the packets to the remote user. All three entity types (sensor node, gateway, and user) are involved in the key agreement scheme, thus, end-to-end security is not provided. Via this approach, the gateway will be an interesting attack vector, since, it could provide access to all data that are sent from all sensor nodes connected to it. We argue that end-to-end security is essential, because, the sensor nodes generate highly sensitive data.

Another approach to an authenticated key agreement scheme in WBANs is presented by Gupta et al. [40]. The scheme assumes three entities: sensor nodes, gateway/user mobile, and authentication server. The scheme requires five steps to securely establish a shared key. However, the authentication server actively participates in the process. This may cause a slow or a potentially impossible connection setup due to the amount of messages and the remote authentication server which may or may not be reachable. In our proposed scheme, we use a distributed authentication approach to enable the patient/user and sensor node to authenticate each other without the involvement of an authentication server.

The work of Chen et al. [19] proposes an anonymous mutual authenticated key agreement scheme for wearable sensor nodes in WBANs. The authors first highlight the shortcomings of another scheme which was proven to be susceptible to the following attacks: offline identity guessing, sensor node impersonation and hub node spoofing attack. Then, they propose an updated scheme that addresses these problems. In their communication model, three entities are considered: second level nodes/sensor nodes, first level nodes/user, and hub nodes/data center. The sensor nodes use this scheme to establish connection with both the user and the data center. However, their proposal is not complete in how the devices are provisioned or updated.

6.2.3 System assumptions, threat model and security requirements

System assumptions

The system model is presented in Figure 6.5. The entities that are considered are the following: Sensor nodes (S), Patient's smartphone (P), and Remote Center (RC). All entities can directly or indirectly communicate with each other, e.g. via an access point or a gateway. The sensor nodes are constrained devices with a wireless interface that are placed on or close to the patient, and, they measure vitals like heart rate and blood pressure. These nodes are divided into groups that represent the selection of sensor nodes required per patient. Furthermore, the data generated by the sensor nodes in one group are collected by the patient on his/her personal device such as a smartphone. The RC is an authorisation server that can validate and authenticate the patient and is able to manage and link the sensor nodes to a patient. Both the patient's device and RC are not considered to be constrained in terms of storage, computation, and energy.

The proof-of-concept implementation that is used for the validation of our proposed scheme consists of one wearable health sensor that is placed on the chest of a patient and monitors vitals such as heart rate and blood pressure

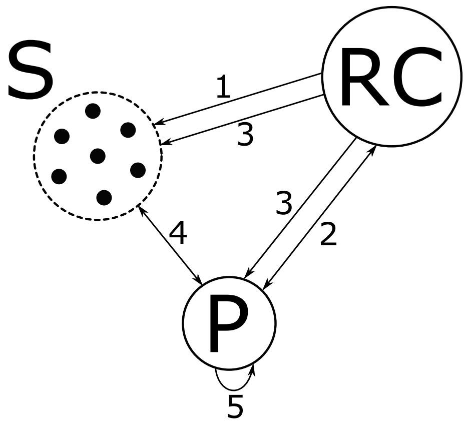


Figure 6.5: System architecture and communication actions used in the proposed scheme, where S denotes the sensor nodes, RC is the remote center, and P is the patient's smartphone.

variation, and communicates via a WiFi wireless interface. This sensor transfers its data to the patient, which is emulated via a Python program on a laptop that is connected to the network.

The consecutive actions between the entities, that will be explained in Section 6.2.4, are indicated in Figure 6.5:

1. Offline installation of the sensor nodes by the remote center;
2. Patient registration with the remote center;
3. Linking of the patient to a group of sensors by the remote center;
4. Key agreement between the sensor nodes and the patient's smartphone;
5. Update of the patient's pin.

Threat model

We assume the Dolev-Yao threat model [31]. An attacker knows the protocol used and may eavesdrop, modify, corrupt, delete, redirect, or replay all the messages that are transmitted. Additionally, this attacker can capture a sensor node and extract all stored parameters from its memory using side-channel attacks. Even in this case, it should not be possible to derive the previously constructed session keys and decrypt the already sent data. The patient's device and RC are assumed to be monitored closely by either the patient or IT staff. The attacker should, therefore, not be able to capture these entities. Furthermore, it is assumed that the RC can communicate securely using classic techniques during the set-up and registration phase. The main focus of our protocol is on providing end-to-end security between the sensor node and the patient's device. However, the goal of the attacker might be to illegitimately obtain the data that are measured by the sensor node, to control the access to the sensor node, or to perform service degradation.

Security requirements

The required cryptographic properties are in line with related work and can be summarised as follows:

- *R.1* *Anonymity and unlinkability:* The identity of the patient for whom the data is collected should not be accessible by an adversary. Also, an adversary should not be able to link multiple sessions to each other that are established between a sensor node and the patient's smartphone.
- *R.2* *Mutual authentication:* Both the sensor node and the patient's smartphone should be able to authenticate each other's identity.
- *R.3* *Perfect forward secrecy:* Previously established sessions keys should not be computable if an adversary acquires the long-term keys of both the sensor node and the patient's smartphone.
- *R.4* *Protection against general attacks:* We consider replay, man-in-the-middle, impersonation, synchronisation, and password guessing attacks as potential attacks.
- *R.5* *Distributed authentication:* The sensor node and patient's smartphone should be able to establish session keys without the intervention of the remote center.

6.2.4 Proposed Solution

Our proposed scheme exists of the following five actions: (1) the remote center (RC) installs the sensor nodes (S) offline, (2) the patient (P) is registered with the RC, (3) the RC links P to S, (4) S and P agree on a key, and (5) P updates the PIN. The overall flow of these actions is presented in Figure 6.5, while more details are provided in Figs. 6.6, 6.7, 6.8, and 6.9. Table 6.4 presents the symbols and their descriptions used throughout the section. First a brief description of the five actions will be given, after which a more detailed explanation is presented.

Table 6.4: Description of the symbols used throughout the section.

Symbol	Description
RC	Remote Center
S_s	Sensor s
ID_i	Identifier of entity i
A_s	Sensor's secret identity
B_s	Sensor's public identity
x_g, y_g, z_g	Secret master keys
PIN_p	Patient's PIN
RPW_p	Patient's password
N	Hash key counter
T_i	Current timestamp
R_i, b	Random value
M_i	Message value
ΔT	Maximum delay
$HREG$	Patient's link
hk	Hash key
sk	Secret key
H	Cryptographic hash function
$\oplus$	XOR operation
$\|\|$	Concatenation operation

- Setup phase: Action 1 consists of the initialisation of the sensor nodes and the RC. This action is done, for example, by the IT staff in a controlled and secure environment.
- Registration phase: At first, the patient needs to register himself/herself to the RC (Action 2). Then, the RC authenticates the patient, remotely configures the sensor nodes, and sends the key that links the patient to the sensor nodes to the patient's smartphone (Action 3).
- Run-time phase: When the sensor nodes are enabled, for example by a nurse that puts the sensor nodes on the patient, they will initiate the key agreement protocol between the sensor node and the smartphone (Action 4).
- Update phase: During the run-time phase, the patient can also choose to update its PIN number using Action 5.

Action 1: Offline installation of Sensor nodes by RC

In Action 1 (shown in Figure 6.6), let x_g, y_g, z_g be three secret master keys chosen by the RC for a group of sensor nodes. Denote ID_s, ID_g the identifiers of respectively a sensor node and a group. The RC computes for each sensor node their secret identity $A_s = \mathsf{H}(ID_s \| ID_g \| x_g)$, public identity $B_s = A_s \oplus \mathsf{H}(y_g)$ and hash key $\mathsf{hk}_s = \mathsf{H}(A_s \| y_g)$. Then, the RC shares the parameters $B_s, \mathsf{H}(A_s), \mathsf{H}(z_g), \mathsf{hk}_s$ with each sensor node in the group via a controlled and secure environment, e.g. a physical channel. Note that the parameter $\mathsf{H}(z_g)$ is the encryption key shared by all sensor nodes in a specific group.

Action 1: Offline installation of sensor nodes by RC

RC		**Sensor**(S_s)
Generates secret master keys: x_g, y_g, z_g for a certain group of sensor nodes.		
Computes for each sensor:		
$A_s = \mathsf{H}(ID_s \| ID_g \| x_g)$		
$B_s = A_s \oplus \mathsf{H}(y_g)$		
$\mathsf{hk}_s = \mathsf{H}(A_s \| y_g)$		
$\mathsf{H}(A_s)$		
$\mathsf{H}(z_g)$		
	$\xrightarrow[\text{physical channel}]{\langle B_s, \mathsf{H}(A_s), \mathsf{H}(z_g), \mathsf{hk}_s \rangle}$	
		Stores B_s
		Stores $\mathsf{H}(A_s)$
		Stores $\mathsf{H}(z_g)$
		Stores hk_s

Figure 6.6: Offline installation of sensor nodes by RC.

Action 2: Patient registration with RC

In Action 2 (shown in Figure 6.7), the patient registers himself/herself using a smartphone via an interface of RC. Let b be a random number chosen by the smartphone. The patient enters/scans his/her identifier ID_p and chooses a personal PIN number PIN_p. The smartphone, then, computes the patient's password $RPW_p = \mathsf{H}(PIN_p \oplus b)$ and key $y_g^* = \mathsf{H}(RPW_p \| ID_p)$. Finally, the smartphone shares ID_p, y_g^* with RC. The RC uses the identifier to validate and authenticate the patient. Upon positive authentication, the RC stores the tuple $\langle ID_p, y_g^* \rangle$.

Action 2: Patient registration with RC

Patient		**RC**
Smartphone generates random value b		
Patient enters ID_p and PIN_p		
$RPW_p = \mathsf{H}(PIN_p \oplus b)$		
$y_g^* = \mathsf{H}(RPW_p \| ID_p)$		
	$\xrightarrow[\text{physical channel}]{\langle ID_p, y_g^* \rangle}$	
		Validates ID_p and stores $\langle ID_p, y_g^* \rangle$

Figure 6.7: Patient registration with RC.

Action 3: Sensor - Patient grouping

Before Action 3 (shown in Figure 6.8), the RC decides which group of sensor nodes will be linked to the patient. Then, the RC computes the hash of the patients identifier $\mathsf{H}(ID_p)$ and the key modifier $\mathsf{H}_{yg}^* = \mathsf{H}(y_g) \oplus y_g^*$. For each sensor, the RC also computes the respective hash key modifier $HK_s = \mathsf{H}(A_s \| y_g) \oplus \mathsf{H}(A_s \| y_g^*)$. Next, the RC shares $\mathsf{H}(ID_p), \mathsf{H}_{yg}^*, HK_s$ securely by encrypting the data with the encryption key $\mathsf{H}(z_g)$. Using these values, the sensor updates its public identity B_s with $B_s^* = B_s \oplus \mathsf{H}_{yg}^*$ and hash key hk_s with $\mathsf{hk}_s^* = \mathsf{hk}_s \oplus HK_s$, and computes and stores the key that links the sensor node group to the patient $HREG = \mathsf{H}(\mathsf{H}(ID_p) \| \mathsf{H}(z_g))$. Furthermore, a counter N_s is created that represents the amount of times the hash key is hashed. After configuring the sensor nodes, the RC also calculates the key $HREG$ and shares it with the patient's smartphone using a secured communication channel.

Action 4: Key agreement between Sensor and Patient

The run-time phase is defined by Action 4. Let R_1, T_1 be a random value and a timestamp generated by the sensor node. Three message parameters are computed: $M_1 = R_1 \oplus \mathsf{H}(A_s)$, $M_2 = \mathsf{H}(HREG \| \mathsf{H}(A_s) \| R_1 \| T_1 \| N_s)$, and $M_3 =$

Action 3: RC links the patient to a group of sensor nodes

RC	**Sensor**(S_s)
Stores $x_g, y_g, z_g, ID_g,$ $[ID_s], ID_p, y_g^*$	Stores $ID_s, B_s,$ $\mathsf{H}(A_s), \mathsf{H}(z_g), \mathsf{hk}_s$

$\mathsf{H}(ID_p)$

$\mathsf{H}_{yg}^* = \mathsf{H}(y_g) \oplus y_g^*$

For each sensor:

$A_s = \mathsf{H}(ID_s \| ID_g \| x_g)$

$HK_s = \mathsf{H}(A_s \| y_g) \oplus \mathsf{H}(A_s \| y_g^*)$

$$\xrightarrow{\mathsf{Enc}_{\{\mathsf{H}(z_g)\}}(\mathsf{H}(ID_p), \mathsf{H}_{yg}^*, HK_s)}$$

$B_s^* = B_s \oplus \mathsf{H}_{yg}^*$

Updates B_s with B_s^*

$HREG = \mathsf{H}(\mathsf{H}(ID_p) \| \mathsf{H}(z_g))$

$\mathsf{hk}_s^* = \mathsf{hk}_s \oplus HK_s$

Updates hk_s with hk_s^*

Creates counter $N_s = 0$

Stores $HREG, \mathsf{H}(ID_p)$

	Patient
	Stores b

$HREG = \mathsf{H}(\mathsf{H}(ID_p) \| \mathsf{H}(z_g))$

Replaces y_g with y_g^*

$$\xrightarrow{\mathsf{Enc}_{y_g^*}\{HREG\}}$$

Stores $HREG$

Figure 6.8: RC links patient to a group of sensor nodes.

$B_s \oplus \mathsf{H}(HREG \| T_1)$. The sensor shares M_1, M_2, M_3, T_1, N_s with the smartphone. Next, the smartphone verifies that T_1 lies within a maximum delay interval ($|T_2 - T_1| < \Delta T$) by generating its own timestamp T_2. Using M_3 and key $HREG$, the public identity of the sensor B_s can be determined. To calculate the secret identity of the sensor node, the smartphone requests the identifier ID_p and PIN PIN_p of the patient. Using these values, the key $y_g^* = \mathsf{H}(\mathsf{H}(PIN_p \oplus b) \| ID_p)$ and the secret identity $A_s = B_s \oplus y_g^*$ can be calculated. Via the secret identity and parameter M_1, the smartphone can compute random value $R_1 = M_1 \oplus \mathsf{H}(A_s)$.

Then, the correctness of A_s, R_1, T_1, N_s is validated using parameter $M_2 \overset{!}{=} M_2^* = \mathsf{H}(HREG \| \mathsf{H}(A_s) \| R_1 \| T_1 \| N_s)$. As a response, the patient's smartphone executes the following actions. If the hash key of this sensor is not yet known by the patient, it can calculate the current hash key using $\mathsf{hk}_s = \mathsf{H}^{N_s}(\mathsf{H}(A_s \| y_g^*))$. Now, with a randomly chosen parameter R_2, the secret encryption key that is to be used for securing the communication between the sensor node and smartphone can be computed via $\mathsf{sk} = \mathsf{H}(R_1 \| R_2 \| T_1 \| T_2 \| B_s \| \mathsf{H}(ID_p) \| \mathsf{hk}_s)$. After the key is generated, the smartphone updates the hash key $\mathsf{hk}_s = \mathsf{H}(\mathsf{hk}_s)$ and generates message parameters $M_4 = R_2 \oplus \mathsf{H}(A_s)$ and $M_5 = \mathsf{H}(\mathsf{sk} \| HREG \| T_1 \| T_2)$. Then, the smartphone shares T_2, M_4, M_5 with the sensor. The timestamp T_2 can be verified using Timestamp T_1 by comparing it to the maximum delay interval $(|T_2 - T_1| < \Delta T)$. Random value R_2 can be computed via $R_2 = M_4 \oplus \mathsf{H}(A_s)$. Next, the sensor can also compute the secret key sk. The correctness of the secret key sk and timestamp T_2 are verified via message parameter $M_5 \overset{!}{=} M_5^* = \mathsf{H}(\mathsf{sk} \| HREG \| T_1 \| T_2)$. Finally, the sensor updates the hash key $\mathsf{hk}_s = \mathsf{H}(\mathsf{hk}_s)$ and increments counter N_s.

Action 5: Patient's PIN update

The patient can also choose to update the PIN during the run-time phase using Action 5, see Figure 6.10. The action's procedure is as follows. Via the smartphone, the patient first enters the old PIN_p and the new PIN_p^*. Then, the smartphone calculates a new value $b^* = PIN_p^* \oplus PIN_p \oplus b$ and replaces the old value with the new value $b = b^*$.

6.2.5 Security analysis

This section provides the detailed security analysis of our proposed scheme according to the security requirements defined in Section 6.2.3. First, the RUBIN formal analysis technique is used to prove that a secure session key is established between the patient's smartphone and a sensor node. We have chosen RUBIN, as it is specifically designed for these types of non-monotonic protocols. Secondly, an informal analysis is given of different popular attacks. Finally, this section compares the offered security features to those of related work.

Action 4: Key agreement between Sensor and Patient

Sensor(S_s)	**Patient**
Stores $ID_s, B_s, \mathsf{H}(A_s), \mathsf{H}(z_g)$ $HREG, \mathsf{H}(ID_p), \mathsf{hk}_s, N_s$	Stores $b, HREG$

Generates random R_1
and timestamp T_1
$M_1 = R_1 \oplus \mathsf{H}(A_s)$
$M_2 = \mathsf{H}(HREG \| \mathsf{H}(A_s) \| R_1 \| T_1 \| N_s)$
$M_3 = B_s \oplus \mathsf{H}(HREG \| T_1)$

$$\xrightarrow{\langle M_1, M_2, M_3, T_1, N_s \rangle}$$

Generates timestamp T_2
Checks $|T_2 - T_1| < \Delta T$
$B_s = M_3 \oplus \mathsf{H}(HREG \| T_1)$
Patient enters ID_p and PIN_p
$y_g^* = \mathsf{H}(\mathsf{H}(PIN_p \oplus b) \| ID_p)$
$A_s = B_s \oplus y_g^*$
$R_1 = M_1 \oplus \mathsf{H}(A_s)$
$M_2^* = \mathsf{H}(HREG \| \mathsf{H}(A_s) \| R_1 \| T_1 \| N_s)$
Checks $M_2^* \overset{!}{=} M_2$
Generates random R_2
If hk_s is not known:
Stores $\mathsf{hk}_s = \mathsf{H}^{N_s}(\mathsf{H}(A_s \| y_g^*))$
$\mathsf{sk} = \mathsf{H}(R_1 \| R_2 \| T_1 \| T_2 \| B_s \| \mathsf{H}(ID_p) \| \mathsf{hk}_s)$
Replaces hk_s with $\mathsf{H}(\mathsf{hk}_s)$
$M_4 = R_2 \oplus \mathsf{H}(A_s)$
$M_5 = \mathsf{H}(\mathsf{sk} \| HREG \| T_1 \| T_2)$

$$\xleftarrow{\langle T_2, M_4, M_5 \rangle}$$

Checks $|T_2 - T_1| < \Delta T$
$R_2 = M_4 \oplus \mathsf{H}(A_s)$
$\mathsf{sk} = \mathsf{H}(R_1 \| R_2 \| T_1 \| T_2 \| B_s \| \mathsf{H}(ID_p) \| \mathsf{hk}_s)$
$M_5^* = \mathsf{H}(\mathsf{sk} \| HREG \| T_1 \| T_2)$
Checks $M_5^* \overset{!}{=} M_5$
Replaces hk_s with $\mathsf{H}(\mathsf{hk}_s)$
Increments N_s

Figure 6.9: Key agreement between Sensor and Patient.

Action 5: Patient's PIN update

Patient(P)

Stores $b, HREG, (\mathsf{sk}, \mathsf{hk}_s, N_s)$

Patient enters PIN_p and PIN_p^*

$b^* = PIN_p^* \oplus PIN_p \oplus b$

Updates b with b^*

Figure 6.10: Procedure to update the PIN of a patient.

Formal verification

Our scheme is verified by using the RUBIN technique that can provide a non-monotonic logic-based verification proof. With RUBIN we can analyse the protocol using a specification that is close to the implementation. Furthermore, we build on the non-monotonic aspect of RUBIN, which means that our protocol can rely on the fact that an entity may no longer possess data it previously knew. Its specification and other examples can be found in [13, 90, 130].

Specification Our scheme assumes that both the patient's smartphone (P) and all sensor nodes (S) trust the Remote Center (RC). Therefore, we focus the proof on the key agreement phase (Action 4). In terms of global sets, the principal set is $PS = \{RC, P, S\}$ where S is the list of sensor nodes S_s in a group. Without loss of generality, we will provide the proof with one of the sensor nodes S_s in the group. The rule set contains the inference rules as defined by Rubin et al. [90]. The secret set is $SS = \{ID_p$, PIN_p, z_g, x_g, y_g, $HREG$, $HK_s\}$ at the start of Action 4. The observer set is as follows:

- Observer$(ID_p) : \{RC, P\}$
- Observer$(\mathsf{H}(ID_p)) : \{RC, S, P\}$
- Observer$(PIN_p) : \{P\}$
- Observer$(z_g) : \{RC\}$
- Observer$(\mathsf{H}(z_g)) : \{RC, P\}$
- Observer$(x_g) : \{RC\}$
- Observer$(y_g) : \{RC\}$
- Observer$(y_g^*) : \{RC, P\}$
- Observer$(HREG) : \{RC, S, P\}$
- Observer$(A_s) : \{RC\}$

- Observer($\mathsf{H}(A_s)$) : $\{RC, S\}$
- Observer(B_s) : $\{RC, S\}$

Now we define the local set for each entity, starting with S_s. Note that the local set of an entity P exists of the Possession Set $POSS(P)$, Belief Set $BEL(P)$, and Behaviour list $BL(P)$.

Principal S_s
$POSS(S_s) = ID_s, B_s, \mathsf{H}(A_s), \mathsf{H}(z_g), \mathsf{H}(ID_p), HREG, \mathsf{hk}_s, N_s$
$BEL(S_s) = \#(HREG), \#(\mathsf{hk}_s), \#(\mathsf{H}(A_s)), \#(\mathsf{H}(z_g)), \#(\mathsf{H}(ID_p)), \#(B_s)$
$BL(S_s) =$

```
SA1  Generate-timestamp($T_1$)
SA2  Generate-secret($R_1$)
SA3  $M_1 \leftarrow R_1 \oplus \mathsf{H}(A_s)$
SA4  $M_2 \leftarrow$ SHA256(Concat($HREG, \mathsf{H}(A_s), R_1, T_1, N_s$))
SA5  $M_3 \leftarrow B_s \oplus$ SHA256(Concat($HREG, T_1$))
SA6  Concat($M_1, M_2, M_3, T_1, N_s$)
SA7  Send($P$, $\{M_1 \cdot M_2 \cdot M_3 \cdot T_1 \cdot N_s\}$)
SA8  Update($\{M_1 \cdot M_2 \cdot M_3 \cdot T_1 \cdot N_s\}$, $\mathcal{W}$)
SA9  Forget($M_1, M_2, M_3$)
SA10 Receive($P$, $\{T_2 \cdot M_4 \cdot M_5\}$)
SA11 Split($\{T_2 \cdot M_4 \cdot M_5\}$)
SA12 Check-freshness($T_2$)
SA13 $R_2 \leftarrow M_4 \oplus \mathsf{H}(A_s)$
SA14 $\mathsf{sk} \leftarrow$ SHA256(Concat($R_1, R_2, T_1, T_2, B_s, \mathsf{H}(ID_p), \mathsf{hk}_s$))
SA15 $M_5^* \leftarrow$ SHA256(Concat($\mathsf{sk}, HREG, T_1, T_2$))
SA16 Check($M_5, M_5^*$)
SA17 $\mathsf{hk}_s \leftarrow$ SHA256($\mathsf{hk}_s$)
SA18 Forget-secret($R_1, R_2$)
SA19 Forget($T_1, T_2, M_4, M_5$)
```

Principal P
$POSS(P) = b, HREG$
$BEL(P) = \#(HREG)$
$BL(P) =$

```
PA1  Receive($S_s$, $\{M_1 \cdot M_2 \cdot M_3 \cdot T_1 \cdot N_s\}$)
```

```
PA2  Split$(\{M_1 \cdot M_2 \cdot M_3 \cdot T_1 \cdot N_s\})$
PA3  Check-freshness$(T_1)$
PA4  $B_s \leftarrow$ SHA256(Concat$(HREG, T_1)$)
PA5  Input$(ID_p, PIN_p)$
PA6  $y_g^* \leftarrow$ SHA256(Concat(SHA256$(PIN_p \oplus b)$, $ID_p$))
PA7  $A_s \leftarrow B_s \oplus y_g^*$
PA8  $\mathsf{H}(A_s) \leftarrow$ SHA256$(A_s)$
PA9  $R_1 \leftarrow M_1 \oplus \mathsf{H}(A_s)$
PA10 $M_2^* \leftarrow$ SHA256(Concat$(HREG, \mathsf{H}(A_s), R_1, T_1, N_s)$)
PA11 Check$(M_2,\ M_2^*)$
PA12 Forget$(M_1, M_2, M_3)$
PA13 Generate-secret$(R_2)$
PA14 Generate-timestamp$(T_2)$
PA15 If $\mathsf{hk}_s \notin POSS(P)$ then $\mathsf{hk}_s \leftarrow$ SHA256$^{N_s}$(SHA256(Concat$(A_s, y_g^*)$)))
PA16 $\mathsf{sk} \leftarrow$ SHA256(Concat$(R_1, R_2, T_1, T_2, B_s, \mathsf{H}(ID_p), \mathsf{hk}_s)$))
PA17 $\mathsf{hk}_s \leftarrow$ SHA256$(\mathsf{hk}_s)$
PA18 $M_4 \leftarrow R_2 \oplus \mathsf{H}(A_s)$
PA19 $M_5 \leftarrow$ SHA256(Concat$(\mathsf{sk}, HREG, T_1, T_2)$))
PA20 Concat$(T_2, M_4, M_5)$
PA21 Send$(S_s,\ \{T_2 \cdot M_4 \cdot M_5\})$
PA22 Update$(\{T_2 \cdot M_4 \cdot M_5\},\ \mathcal{W})$
PA23 Forget-secret$(R_1, R_2, ID_p, PIN_p, A_s, \mathsf{H}(A_s))$
PA24 Forget$(T_1, T_2, M_4, M_5, B_s)$
```

Analysis The scheme starts with the execution of the SA1-SA9 RUBIN actions of $BL(S_s)$, resulting in the updated local set of principal S_s described below. Also, the Update action results in Observers(M_1, M_2, M_3, T_1, N_s) $= \mathcal{W}$. Principal S_s generates message parameters M_{1-3} that are believed to be fresh because of the freshness believes of terms R_1, T_1, and $HREG$. After sending the message, principal S_s discards M_1, M_2, M_3 in SA9, since, they are no longer needed.

$POSS(S_s) = ID_s, B_s, \mathsf{H}(A_s), \mathsf{H}(z_g), HREG, \mathsf{H}(ID_p), \mathsf{hk}_s, N_s, T_1, R_1, M_1, M_2,$
M_3
$BEL(S_s) = \#(HREG), \#(\mathsf{hk}_s), \#(\mathsf{H}(A_s)), \#(T_1), \#(R_1), \#(M_1), \#(M_2),$

$\#(M_3)$
$BL(S_s) = SA1, ..., SA8$

The next four RUBIN actions to be executed are those of principal P, because, the terms $\{M_1 \cdot M_2 \cdot M_3 \cdot T_1 \cdot N_s\}$ are sent to it. The new local set described below is the result of this. The terms M_{1-3}, N_s are not believed to be fresh, because, they cannot directly be verified. The freshness of timestamp T_1, however, is ensured using the Check-freshness action.

$POSS(P) = b, HREG, N_s, T_1, M_1, M_2, M_3, B_s$
$BEL(P) = \#(HREG), \#(T_1),\ \{\{S_s, P, RC\} \subseteq \text{Observer}(HREG)\}$
$BL(P) = PA1, \ldots, PA4$

The next RUBIN action collects the input ID_p and PIN_p of the patient, but, it can not be verified to be fresh. Then, the following six RUBIN actions of principal P are executed. They results in the new local set described below. By checking $M_2 \overset{!}{=} M_2^*$, Observer($\mathsf{H}(A_s), R_1$)=$\{S_s, P\}$ is proven, since, $\{S, P, RC\} \subseteq$ Observer($HREG$) and Observer($\mathsf{H}(A_s)$) $= \{RC, S_s, P\}$.

$POSS(P) = b, HREG, N_s, T_1, M_1, M_2, M_3, B_s, y_g^*, A_s, \mathsf{H}(A_s), R_1, ID_p, PIN_p$
$BEL(P) = \#(HREG), \#(T_1),\ \#(A_s),\ \#(R_1)$
$BL(P) = PA1, \ldots, PA11$

The PA12-PA22 RUBIN actions of principal P are performed if the verification of M_2 is successful. The local set described below is the result. The secret and hash key are generated and are believed to be fresh via respectively $\#(A_s)$ and $\#(R_1, R_2, T_1, T_2, \mathsf{hk}_s)$. Then, principal P calculates message parameters M_4 and M_5.

$POSS(P) = b,\ HREG,\ (\mathsf{hk}_s, N_s),\ T_1,\ B_s,\ y_g^*,\ A_s,\ \mathsf{H}(A_s),\ R_1,\ ID_p,\ PIN_p,\ R_2,\ T_2,\ M_4,\ M_5,\ \mathsf{sk}$
$BEL(P) = \#(HREG),\ \#(T_1),\ \#(A_s),\ \#(R_1),\ \#(R_2),\ \#(T_2),\ \#(\mathsf{sk}),\ \#(\mathsf{hk}_s),\ \#(M_4),\ \#(M_5)$
$BL(P) = PA1, \ldots, PA22$

The last RUBIN action of P is to forget all terms computed or generated in this process except for (hk_s, N_s) and sk via RUBIN actions $PA23$ and $PA24$.

Principal P sends $\{T_2 \cdot M_4 \cdot M_5\}$ to principal S_s which enables the sensor node to continue executing RUBIN actions SA10-SA17. The resulting local set is as provided below. After splitting the received terms, the freshness of timestamp T_2 is verified. Via Check(M_5, M_5^*), S_s can believe Observer(sk) $= \{S_s, P\}$. Furthermore, the freshness of sk is ensured via the random values and the time stamps. Next, the hash key hk_s^i is updated with hk_s^{i+1}, thus, POSS(S_s) := POSS(S_s) - hk_s^i + hk_s^{i+1}.

$POSS(S_s) = ID_s$, B_s, $\mathsf{H}(A_s)$, $\mathsf{H}(z_g)$, $HREG$, $\mathsf{H}(ID_p)$, hk_s, N_s, T_1, R_1, T_2, R_2, M_4, M_5, sk
$BEL(S_s) = \#(HREG), \#(\mathsf{hk}_s), \#(\mathsf{H}(A_s)), \#(T_1), \#(R_1), \#(T_2), \#(\mathsf{sk})$
$BL(S_s) = SA1, \ldots, SA17$

Finally, principal S_s forgets all terms computed or generated in this process except for (hk_s, N_s) and sk via RUBIN actions $SA18$ and $SA19$. The observer set is as follows at the end of Action 4:

- Observer(ID_p) : $\{RC, P\}$
- Observer$(\mathsf{H}(ID_p))$: $\{RC, S, P\}$
- Observer(PIN_p) : $\{P\}$
- Observer(z_g) : $\{RC\}$
- Observer$(\mathsf{H}(z_g))$: $\{RC, P\}$
- Observer(x_g) : $\{RC\}$
- Observer(y_g) : $\{RC\}$
- Observer(y_g^*) : $\{RC, P\}$
- Observer$(HREG)$: $\{RC, S, P\}$
- Observer(A_s) : $\{RC\}$
- Observer$(\mathsf{H}(A_s))$: $\{RC, S\}$
- Observer(B_s) : $\{RC, S\}$
- Observer(sk) : $\{S, P\}$
- Observer(hk) : $\{S, P\}$

This analysis implies that:

- R_1, T_1, R_2, T_2 are fresh for each session and are known by the legitimate S_s and P. This ensures resilience against replay attacks and linkability.
- S_s and P are mutually authenticated, protecting against man-in-the-middle and impersonation attacks.
- ID_p and A_s are only possessed by P and RC, providing anonymous communication.
- The secret key sk is only known to S_s and P and is an independent key for each session as it is computed over random values and the hash key hk_s. Note that all consecutive hash keys can only be known at any time by P and RC, thus, the scheme achieves perfect forward secrecy if a sensor node is compromised.

- The random value b and a scan based ID_p, which is a random identifier generated by the hospital, are used to protect against password guessing attacks.

Informal verification

The security verification is continued in this subsection with an informal discussion of the provided security properties and the protection against potential security attacks. Note that we assume the threat model described in Section 6.2.3.

Patient impersonation attack: Suppose an attacker $\mathcal{A}$ tries to create valid input credentials. The attacker needs to enter an identifier and PIN code (ID_p^a, PIN_p^a) on the patient's smartphone, assuming that $\mathcal{A}$ can get access to the smartphone. $\mathcal{A}$ cannot determine the correct values via eavesdropping the communication, because, they are not shared in plain text nor over an insecure channel. Furthermore, ID_p cannot be computed from $\mathsf{H}(ID_p)$ or $HREG = \mathsf{H}(\mathsf{H}(ID_p)||\mathsf{H}(z_g))$, which are stored on the sensor nodes, since, we consider the hash function a one-way function.

Sensor node impersonation attack: An attacker $\mathcal{A}$ intercepts the first transmission $\langle M_1, M_2, M_3, T_1, N_s \rangle$ of sensor node S_s and tries to generate a valid message $\langle M_1^a, M_2^a, M_3^a, T_1^a, N_s^a \rangle$ by impersonating this node. $\mathcal{A}$ can generate a random value R_1^a, timestamp T_1^a, and hash key counter N_s^a. However, $\mathcal{A}$ cannot generate valid message parameters M_1, M_2, or M_3, because, this require the knowledge of secret values $\mathsf{H}(A_s)$ and $HREG$. Moreover, these secret values cannot be guessed in polynomial time. Message parameter M_1 protects $\mathsf{H}(A_s)$ via the one-time pad technique, and, message parameters M_2 and M_3 protect $HREG$ via the one-way hash function.

Replay attack: Assume an attacker $\mathcal{A}$ has eavesdropped on the messages that are sent during the key establishment procedure (Action 4) and tries to replay these messages to either the patient's smartphone or sensor node. The freshness of both messages are easily verifiable via the timestamps T_1 and T_2, also, the integrity of the timestamp is guaranteed via M_2 and M_5.

Eavesdropping attack: Messages sent over an insecure channel can be eavesdropped by an attacker $\mathcal{A}$, thus, messages $\langle M_1, M_2, M_3, T_1, N_s \rangle$ and $\langle T_2, M_4, M_5 \rangle$ are seen by $\mathcal{A}$. The secret key sk is generated via $\mathsf{sk} =$

$\mathsf{H}(R_1\|R_2\|T_1\|T_2\|B_s\|\mathsf{H}(ID_p)\|\mathsf{hk}_s)$. $\mathcal{A}$ cannot derive R_1, R_2, B_s, $\mathsf{H}(ID_p)$, and hk_s, since, these values are protected via either the one-time pad technique or the one-way hash function.

Privileged insider attack: Assume an adversary $\mathcal{A}$ who may be a privileged sensor node that is connected to patient P_i and that tries to establish a connection with another patient's smartphone P_j. $\mathcal{A}$ cannot compute the sensor node-patient link $HREG_j$, thus, he cannot create a valid message parameter M_2. Upon validation, the patient's smartphone will abort the key establishment procedure. However, $\mathcal{A}$ can determine the other sensor nodes that are linked to P_i by eavesdropping on the communication. Using the value $HREG_i$ and the eavesdropped message parameter T_1, the identity B_s of all linked sensor nodes can be computed via $B_s = M_3 \oplus \mathsf{H}(HREG_i\|T_1^a)$.

Offline password guessing attack: In our threat model, an attacker $\mathcal{A}$ can retrieve the stored secrets of a sensor node via side-channel attacks. Within the time frame of the envisioned sessions of a patient in a hospital, $\mathcal{A}$ should not be able to guess the identifier ID_p or PIN_p via the values $H(ID_p)$ and B_s where $B_s = A_s \oplus \mathsf{H}(y_g)$ and $y_g = \mathsf{H}(\mathsf{H}(PIN_p \oplus b)\|ID_p)$.

Anonymity and unlinkability: The identity of the patient ID_p is masked in our scheme via the one-way hash function. For an attacker $\mathcal{A}$, it should be computationally infeasible to compute it. Furthermore, unlinkability is also provided via the freshness of random values and timestamps. Suppose $\mathcal{A}$ can intercept messages from the sensor node and the patient's smartphone. Using only an eavesdropping attack, the attacker should not be able to compute B_s, $\mathsf{H}(ID_p)$, $\mathsf{H}(A_s)$, or $HREG$ from one of the message parameters within polynomial time.

Perfect forward secrecy: Assume an attacker $\mathcal{A}$ compromises the session key sk, the sensor node's secret and public identity $\mathsf{H}(A_s), B_s$ and the current hash key hk_s^i, and the patient's link $HREG$. These keys do not reveal any previous session keys as session keys are calculated as $\mathsf{sk} = \mathsf{H}(R_1\|R_2\|T_1\|T_2\|B_s\|\mathsf{H}(ID_p)\|\mathsf{hk}_s^i)$. The previous hash key hk_s^{i-1} is not computable, since, during each session establishment the hash key is updated via $\mathsf{hk}_s^i = \mathsf{H}(\mathsf{hk}_s^{i-1})$. Moreover, the freshness of the session keys is ensured through the use of the timestamps, and the random values.

Comparison to related work

An overview of the security features that our work and the considered related work provide is presented in Table 6.5. Only our scheme covers all the required security features. Gupta et al. [40] and Shuai et al. [102] do not provide distributed key agreement, and, Gupta et al. [40], Kumar et al. [57], and Chen et al. [19] do not provide sufficient perfect forward secrecy protection. Furthermore, if we take the requirement to input an identifier and a password into account, our scheme and that of Gupta et al. and Shuai et al. [102] would only be eligible, because, the schemes of Kumar et al. [57] and Chen et al. [19] do not provide this feature.

Table 6.5: Comparison of the available security features. The security features R.1-R.5 are described in Section 6.2.5 (R.1: Anonymity and unlinkability, R.2: Mutual authentication, R.3: Perfect forward secrecy, R.4: Protection against general attacks, and R.5: Distributed authentication).

Security features	**R.1**	**R.2**	**R.3**	**R.4**	**R.5**
Kumar2017 [57]	Y	Y	N	Y	Y
Gupta2019 [40]	Y	Y	N	Y	N
Chen2018 [19]	Y	Y	N	Y	Y
Shuai2019 [102]	Y	Y	Y	Y	N
Ours	**Y**	**Y**	**Y**	**Y**	**Y**

6.2.6 Implementation results

The implementation results are generated using a generic and an empirical study. The computationally expensive operations are identified and generalised for each entity of our scheme and related key agreement protocols. Furthermore, the communication impact is compared using the amount of messages and bits that are exchanged. Next, the impact of the proposed scheme is validated empirically through a proof-of-concept implementation.

The computation impact of the key agreement protocol is compared to related work in Table 6.6 in terms of the number of hash functions (Th), the number of symmetric-key encryptions (Te), the number of symmetric-key decryptions (Td), and the number of fuzzy key extractions (Tfe). Our scheme requires five and nine hash operations for respectively the sensor node and the patient. Also, our scheme does not require input of the RC during key agreement. In total, our scheme ranks third in terms of performance. Only Chen et al. [19] and Kumar et al. [57] feature less operations.

Table 6.6: Comparison of the computation impact of the key agreement scheme with other relevant state-of-the-art schemes (Th: hash, Te: symmetric encryption, Td: symmetric decryption, and Tfe: fuzzy extraction).

Protocol	**Sensor**	**User**	**RC**	**Total**
Kumar2017 [57]	2Th + 1Te + 1Td	2Th + 1Td + 2Te	-	4Th + 3Te + 2Td
Gupta2019 [40]	4Th	7Th	5Th	16Th
Chen2018 [19]	5Th	8Th	-	13Th
Shuai2019 [102]	7Th	Tfe + 11Th	12Th	Tfe + 30Th
Ours	5Th	9Th	-	14Th

The communication impact compared to related work is given in Table 6.7. We have chosen for the following message parameter sizes: 24-byte ID, 32-byte hash digest (SHA256), 32-byte random, 4-byte timestamp (Unix epoch), and 4-byte counter. The results of Kumar et al. [57] and Chen et al. [19] were estimated because the original value was either not available or inconsistent. Our scheme features the lowest amount of bits exchanged. Furthermore, only two messages rounds are required, which is equal for the protocol of Kumar et al. [57] and Chen et al. [19].

Table 6.7: Comparison of the communication impact of the key agreement scheme with other relevant state-of-the-art schemes (*: estimated values).

Protocol	**Rounds**	**Total size (bits)**
Kumar2017 [57]	2	1568*
Gupta2019 [40]	5	3808
Chen2018 [19]	2	2080*
Shuai2019 [102]	5	1824
Ours	2	1376

An empirical study of our proposed scheme is done using a proof-of-concept implementation on a custom-designed board featuring a TI MSP432P4011 MCU [109] and a TI CC3120 network processor [108]. The MCU is a 32-Bit Arm Cortex-M4F running at 48 Mhz with 2 MB of Flash and 256 KB of RAM. The code is compiled using the GNU GCC compiler tools version 7.2.1 with the -O3 optimisation setting. Note that this board is the same as the one used in the evaluation of Section 6.1. The performance of Action 4 is measured on this health sensor node using the available 24-bit system timer, and, 75 iterations were performed. The health sensor node communicates wirelessly with a Patient that is emulated using python code running on a Dell XPS 15 9570 laptop. The results are presented using a box plot and a 96% confidence interval in Figure 6.11. On average, Action 4 takes about 26.5 ms. Using the power consumption of the microcontroller and the network interface, the energy usage is estimated to be about 507 μJ. The energy consumption of the key establishment of our proposed scheme is negligible in comparison to the energy required to perform the measurements and communication of sensor data in one 10 s period, which is estimated to be around 698.2 mJ.

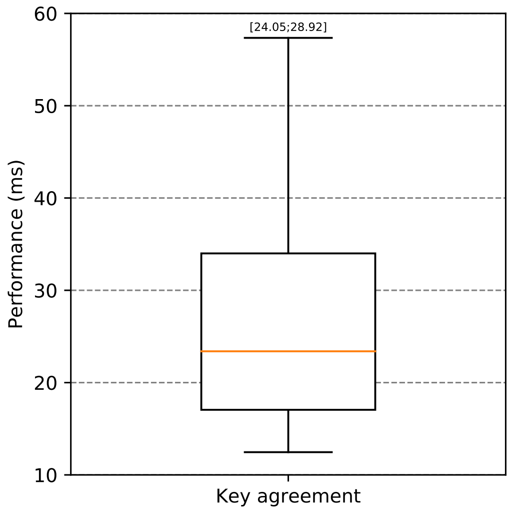


Figure 6.11: Performance box plot of Action 4 based on 75 runs.

6.2.7 Conclusion

This work proposes the design and implementation of a symmetric-key security protocol for wearable healthcare devices. Both the association phase, in which the wearable device is anonymously linked to the patient, and the communication phase, in which end-to-end secured data exchange takes place, are described in detail. An analysis of the security features and the communication and computation cost of the protocol shows that our solution outperforms previously proposed protocols. In terms of future work, one additional security feature is worth considering, namely the protection against privileged insider attacks, in which the adversary can determine the identity of the sensor nodes that are linked to a patient.

6.3 Comparison of the two solutions

First, end-to-end security is provided for a mobile health system using public-key cryptography. The system is evaluated using a proof-of-concept implementation that can be used in a hospital scenario. The TLS protocol using the DHE_RSA_AES_GCM cipher suite is chosen, providing a broad range of security guarantees. Also, it can be offloaded to the network processor on the mobile health system, leading to a lower energy consumption. The results show that a security session can be established using about 32 mJ, while, it uses a negligible amount of storage and memory. Though, it requires about 230 ms to establish a session and around 4567 B of data to be communicated. The amount of data that TLS requires to establish a connection is measured by analysing the traffic between the sensor node and the local server during session establishment.

Secondly, a security protocol is designed that is based on symmetric keys and provides a similar security level. It is evaluated through a proof-of-concept implementation. In comparison to the system secured using public-key cryptography, it additionally needs a authorisation server, and, the patient's smartphone will receive the sensor data instead of a local server. The resulting protocol requires only around 507 μJ, 26 ms and 172 B of data to establish a security session. Though, additional firmware is required to implement it.

We compare the results of both approaches in this paragraph. The advantage of the symmetric-key cryptography based protocol is its efficiency in terms of energy cost, performance, and data communication. Its disadvantage is the need of an authorisation server to configure each sensor node for each patient. Additionally, the authorisation server needs to maintain the shared keys of each sensor node. In contrast, in the public-key based protocol, the authentication of entities is enabled through the use of certificates. If two entities hold a valid certificate and key pair, they can securely communicate with each other. Authorisation is achieved via login credentials. Furthermore, a hardware accelerator is used to offload the security protocol. It results in a negligible overhead in terms of storage and memory. However, it requires much more energy, time and data communication to establish a security session using the public-key cryptography based protocol in comparison to the symmetric-key solution.

6.4 Conclusion

We propose two security protocols and proof-of-concept implementations, to be used in energy-constrained mobile health systems. The first solution is based on public-key cryptography and makes use of existing cipher suites. Our contribution consists of an evaluation of the energy consumption of both the computations in the cipher suites and the wireless communication protocols. Moreover, we implement the proposed solution on a real-life medical prototype measuring five different vital sign parameters. The second solution proposed in this chapter, relies on symmetric-key cryptography and proposes a novel protocol for end-to-end security in the specific setting where the smartphone of the patient collects all sensor data. Both a formal and an informal proof of the security properties is given. Finally, a comparison is made between the two schemes, based on the energy consumption, the system setup and the security features.

Chapter 7

Conclusion

In this thesis, the following four challenges are considered: (1) securing heterogeneous devices and networks, (2) enabling digital signatures on constrained IoT devices, (3) analysing the energy consumption of security protocols, and (4) providing security solutions for mobile health systems. First, conclusions for each challenge are presented. Then, potential future work is discussed.

7.1 Conclusions

The HeComm architecture was proposed to tackle the end-to-end security challenge in a heterogeneous IoT environment in Chapter 3. Via the analysis of the proof-of-concept that interconnected a 6LoWPAN and a LoRaWAN network, object security and fog computing proved to be optimal concepts for this purpose. Object security reduced the compatibility concerns of different IoT networks while providing the required security guarantees. Furthermore, fog nodes provided the interconnection of IoT networks, and could not compromise the security of the communication between the IoT nodes in the HeComm architecture.

The storage optimisations for the coupon technique, which were discussed in Chapter 4, proved to be ideal to enable digital signatures for constrained IoT devices. Also, the proposed encoding and storage encryption ensured the usability of the coupon technique. Overall, the coupon technique along with the proposed storage optimisations reduced the impact of digital signature generation in terms of performance and energy consumption. However, the storage requirement of this technique remains a challenge. This was partially solved by using the coupon addition technique where coupons were added to each other to create new coupons. To clarify, combinations of coupons were used to enlarge the usable set of coupons while only storing a few. The performance impact of a coupon addition is considerable in comparison to a signature encryption using the coupon technique, but, it is about six times more efficient than a signature encryption without the coupon technique. The downside of coupon addition is that it does not scale well, since, a combination of n coupons requires $n - 1$ coupon additions. Consequently, a limited number of coupons should be used in a combination to limit the performance impact.

Our analysis of the energy consumption of security protocols in Chapter 5 can serve as a guideline for practitioners and researchers selecting the appropriate security algorithms and wireless communication protocols. Additionally, an equation to calculate the minimal time period of a security/communication session was introduced that can limit the impact of the session setup procedure of security protocols on the overall energy consumption. In summary, we used a granular approach to estimate the energy consumption of security algorithms and protocols. The granular approach provided a rough estimation, nevertheless, more experiments are required to validate the accuracy of it.

In chapter 6, we analysed two different solutions to provide end-to-end security for mobile health systems: via public-key cryptography and via symmetric-key cryptography. The public-key approach provided strong security guarantees but expressed a considerable impact in terms of performance and energy. However, the impact was limited through the use of hardware accelerators, and by limiting the refresh rate of the security session. The solution based on symmetric-key cryptography, on the other hand, was designed to provide strong and efficient key establishment. Its impact in terms of performance and energy consumption was considerably lower than the public-key based solution, while, providing similar security guarantees. Note that the symmetric-key solution was specifically designed to fit the requirements of the mobile health system, whereas, the public-key solution focused on the use of known and proven protocols. In small to medium scale applications like the hospital scenario addressed in this thesis, the symmetric-key solution might be the best fit, because of its efficiency. Though, public-key based solutions using a public key infrastructure may be more suitable for large scale applications, where e.g. thousands of devices are used, as it does not require the upkeep of the shared secrets for each device.

7.2 Future work

This thesis focuses on the in-depth and practical study of the aforementioned security challenges to provide energy-efficient and performance-aware end-to-end security in a heterogeneous IoT environment. While, the identified techniques, analyses, and proposed architectures can be enhanced further upon, the focus should be on the adoption of the identified techniques, as they can serve as a foundation to build industry-ready solutions.

The COSE standard that is used in the HeComm architecture should be updated, in the future, with the OSCORE standard. OSCORE also uses the COSE specification, but, it additionally provides authentication guarantees to some header fields of the CoAP protocol. Secondly, more fog interfaces should be devised to support other IoT networks as well. For the heterogeneous challenge (1) in general, the difficulty lies in the interoperability of legacy and newly-developed systems. New systems can be designed with interoperability in mind, i.e. using standardised protocols to communicate. Legacy systems, on the other hand, are typically never updated. Future work could look, for example, at how to provide end-to-end security to legacy systems in a heterogeneous environment with minimal adaptations.

We proposed the coupon technique to enable digital signatures on constrained IoT devices (2). The storage requirements of this technique were optimised, though, the impact in terms of storage and computation is still considerable. Also, the lifetime of these devices should not be limited to the number of signatures the device can compute. Consequently, we could derive the following research question. How to keep the IoT device running while using a limited supply of coupons? Future research could look at e.g. updating the coupons on-the-fly or devising a flexible security architecture to handle this.

In this thesis an analysis is given on the energy consumption of security protocols (3) which can serve as a guideline. It is done by identifying the required cryptographic operations and estimating the energy consumption using a granular approach. Our process consisted of analysing the specifications and source code and combining all the results. Since it is a reasonably tedious process, it has only been applied to a select number of protocols. Furthermore, this process requires a cryptographic background. In the future, research could focus on creating development tools that estimate the energy consumption of security architectures using the provided source code/diagram.

Finally, two end-to-end security solutions were devised for mobile health systems (4). However, the implementation of the key infrastructure was not considered. The discussed systems were small to medium-scale systems, thus, the devices can easily be provisioned and maintained by e.g. the local IT department. In large-scale systems, on the other hand, the bootstrapping phase, i.e. management of the keys and configuration, should be taken into account. This raises the following research question. How to manage the key infrastructure of security architectures semi-automatically with a limited amount of user interaction in large-scale and low-power IoT systems?

Overall, future research should focus on optimising existing or inventing new standards to enhance the adoption of energy-efficient security architectures in IoT systems.

Appendix A

Workflow of medical sensor system in a hospital

The workflow describes how the patch is used within the involved hospitals, as depicted in Figure A.1. At first, every patch needs to be configured. This includes setting up the network *configuration* and security parameters for the connection to the hospital network, and registering the device into the system database. The patch configuration as well as future *updates* are done by the IT department of the hospital using a programmer with a wired connection to a PC.

After configuration, the patch is delivered to the clinical staff. Note that any user that wants access to the system first needs to go through an *authentication* step. This influences which actions he/she can perform with the system and how the data are filtered on the display.

During the *association* step, a given patch is associated with a patient. Each time a patch is assigned to a patient, the nurse scans the patient's EMR identifier on the patient's wristband and the patch serial number on the patch's label using an off-the-shell barcode reader. Based on these two identifiers, a unique association identifier (UID) is generated by the local server. This UID is wirelessly sent from the local server to the patch each time it boots up for the duration of the association. The patch will append the UID to each packet it will subsequently emit. This design has two main advantages:

- It does not expose the patient's EMR identifier inside the message broker.

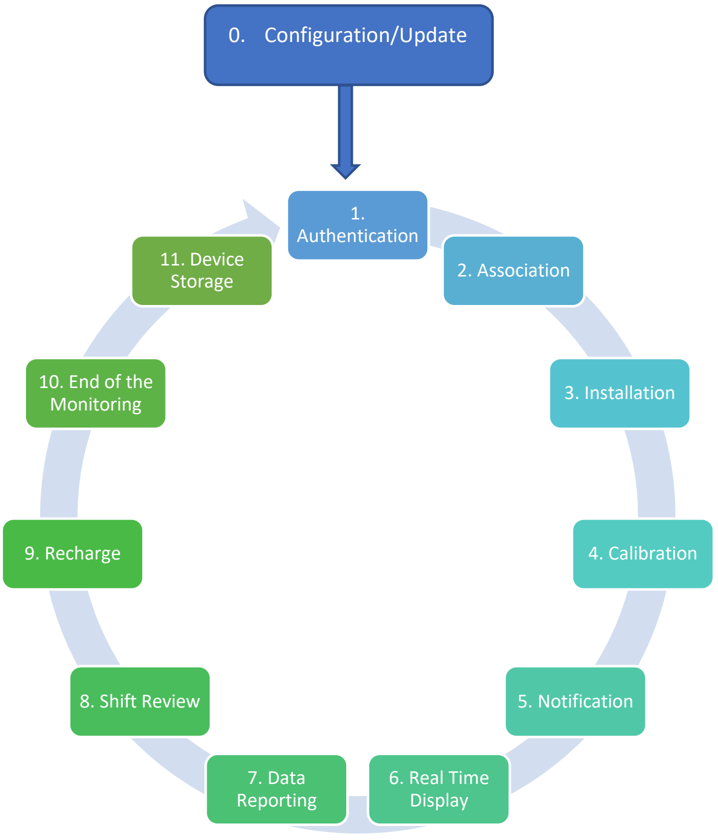


Figure A.1: The workflow of the wearable health system visualised in a flow diagram.

- It avoids any session mismatch in the data processing pipe-lines and allows the pipelines to cache UIDs and related metadata for efficiency.

The nurse will now *install* the patch on the patient's chest. First, he/she checks the battery level of the patch, second, he/she attaches the patch with an adhesive on the left chest at the level of the heart, and third, he/she attaches both electrodes to the patch via snap buttons and to the chest using the adhesive part of the electrodes. All vital parameters of the patient except the blood pressure are now displayed continuously on the user interface.

It is then required to *calibrate* the patch for the blood pressure measurement. A reference blood pressure measurement is taken by the nurse and sent to the blood pressure measurement algorithm via the user interface.

All vital parameters of the patient are now monitored in (near) real time and the patient's early warning score (EWS) is continuously evaluated. The *real-time display* can be accessed on any web browser or mobile device on the hospital's network.

If a deviation is detected in the vital parameters, a *notification* is sent to the clinical staff. The clinical staff can comment and/or acknowledge this notification. Furthermore, all actions are tracked by the local server. This offers an audit log and a communication channel between clinical staff. With the right permissions, a user can adjust the thresholds of the notification detection for a specific patient.

The clinical staff can report specific data points to the EMR. The *data reporting* mimics the work they were already doing with discrete measuring devices. However, compared to the discrete measurement approach, the process is faster via our interface, and, it is more detailed because the clinical staff is able to select more data points.

At the end of a staff member's shift, he/she can *review* the notifications and the evolution of vital parameters. He/she can also review and export a PDF report to the EMR of any ECG sample.

For the entire duration of the session, the battery charge level will be visible on the user interface. If the battery is almost empty, the clinical staff has to replace the battery. The system will notify the clinical staff whenever the battery level drops below 20 and 5 percent, such that a timely *recharge* can be done.

When the patient no longer needs to be monitored, the adhesive is thrown away and the patch is put back in the stock after cleaning. This refers to the *end of monitoring* and the *device storage* steps in Figure A.1. The clinical staff instructs the system that the patch is now available to be used for another patient. Note that this will happen automatically if no data are received for a long period of time.

Bibliography

[1] 802.11 Working Group of the LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Information technology —Telecommunications and information exchange between systems Local and metropolitan area networks — Specific requirements. Tech. rep., IEEE, New York, NY, USA, 2016.

[2] Aranha, D. F., and Gouvêa, C. P. L. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`. [Online; accessed November-2019].

[3] Arm Limited. ARM v7-M Architecture Reference Manual. `https://static.docs.arm.com/ddi0403/eb/DDI0403E_B_armv7m_arm.pdf`, 2014.

[4] Arm Limited. Arm Mbed TLS. `https://tls.mbed.org/`, 2019. [Online; accessed June-2019].

[5] Arm Limited. GNU Arm Embedded Toolchain. `https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm`, 2019. [Online; accessed October-2019].

[6] Ashton, K. That ‘internet of things’ thing. *RFID journal 22*, 7 (2009), 97–114.

[7] Atzori, L., Iera, A., and Morabito, G. The Internet of Things: A survey. *Computer Networks 54*, 15 (Oct. 2010), 2787–2805.

[8] Barker, E. B., and Kelsey, J. M. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, June 2015.

[9] Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. The Keccak reference. Tech. rep., STMicroelectronics, NXP Semiconductors, 2011.

[10] Bluetooth, SIG. Core system package [low energy controller volume]. *Specification of the Bluetooth System v4.1 6* (2013), 2467–2640.

[11] Bormann, C., Ersue, M., and Keränen, A. Terminology for Constrained-Node Networks. RFC 7228, 2014.

[12] Bormann, C., and Hoffman, P. RFC 7049: Concise Binary Object Representation (CBOR). Tech. rep., Internet Engineering Task Force (IETF), Oct. 2013.

[13] Braeken, A., Liyanage, M., Kumar, P., and Murphy, J. Novel 5G Authentication Protocol to Improve the Resistance Against Active Attacks and Malicious Serving Networks. *IEEE Access 7* (2019), 64040–64052.

[14] Braeken, A., Shabisha, P., Touhafi, A., and Steenhaut, K. Pairing free and implicit certificate based signcryption scheme with proxy re-encryption for secure cloud data storage. In *2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)* (2017), IEEE, pp. 1–7.

[15] Brorsson, J., and Gunnarsson, M. Compact object security for the internet of things, 2016. Student Paper.

[16] Cabo. cn-cbor: A constrained node implementation of CBOR in C. `https://github.com/cabo/cn-cbor`, 2013. [Online; accessed November-2019].

[17] Certicom Research. SEC 1: Elliptic curve cryptography. *Standards for efficient cryptography 2* (2009).

[18] Certicom Research. SEC 4: Elliptic curve qu-vanstone implicit certificate scheme (ECQV). *Standards for efficient cryptography 1* (2013).

[19] Chen, C.-M., Xiang, B., Wu, T.-Y., and Wang, K.-H. An Anonymous Mutual Authenticated Key Agreement Scheme for Wearable Sensors in Wireless Body Area Networks. *Applied Sciences 8*, 7 (jul 2018), 1074.

[20] Cheong, P. S., Bergs, J., Hawinkel, C., and Famaey, J. Comparison of LoRaWAN classes and their power consumption. In *2017 IEEE Symposium on Communications and Vehicular Technology (SCVT)* (Nov. 2017), vol. 2017-Decem, IEEE, pp. 1–6.

[21] Cirani, S., Ferrari, G., Iotti, N., and Picone, M. The IoT hub: A fog node for seamless management of heterogeneous connected smart objects. In *2015 12th Annu. IEEE Int. Conf. Sensing, Commun. Netw. - Work. SECON Work. 2015* (2015), pp. 43–48.

[22] Claes, S., Berckmans, D., Geris, L., Myin-Germeys, I., Audenhove, C. V., Diest, I. V., Hoof, C. V., Hoyweghen, I. V., Huffel, S. V., and Vrieze, E. Mobile Health Revolution in Healthcare: Are We Ready? Tech. rep., Metaforum KU Leuven, Leuven, 2019.

[23] Čolaković, A., and Hadžialić, M. Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. *Computer Networks 144* (Oct. 2018), 17–39.

[24] Contiki. Contiki: The Open Source OS for the Internet of Things. `www.contiki-os.org`. [Online; accessed 2017].

[25] Cose-wg. Implementation of COSE in C using cn-cbor and openssl. `https://github.com/cose-wg/COSE-C`, 2017.

[26] Daemen, J., and Rijmen, V. The block cipher rijndael. In *International Conference on Smart Card Research and Advanced Applications* (1998), Springer, pp. 277–284.

[27] Dang, Q. H. Secure Hash Standard. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, July 2015.

[28] de Clercq, R., Uhsadel, L., Van Herrewege, A., and Verbauwhede, I. Ultra Low-Power implementation of ECC on the ARM Cortex-M0+. In *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference - DAC '14* (New York, New York, USA, 2014), ACM Press, pp. 1–6.

[29] Diffie, W., and Hellman, M. New directions in cryptography. *IEEE transactions on Information Theory 22*, 6 (1976), 644–654.

[30] Docker Incorporated. Docker enterprise is the industry-leading container platform. `https://www.docker.com/products/docker-enterprise`, 2019. [Online; accessed March-2019].

[31] Dolev, D., and Yao, A. On the security of public key protocols. *IEEE Transactions on Information Theory 29*, 2 (mar 1983), 198–208.

[32] Dworkin, M. J. Recommendation for block cipher modes of operation. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, 2001.

[33] Dworkin, M. J. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, 2007.

[34] Dworkin, M. J. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, July 2015.

[35] em microelectronics. EM9301 - Single-Cell Battery Bluetooth Low Energy Controller. `https://www.emmicroelectronic.com/product/standard-protocols/em9301`, 2018. [Online; accessed June-2019].

[36] Gandolfi, K., Mourtel, C., and Olivier, F. Electromagnetic analysis: Concrete results. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (2001), C. K. Koc, D. Naccache, and C. Paar, Eds., no. 2162 in Lecture Notes in Computer Science, Springer-Verlag, p. 255–265.

[37] Giry, D. Cryptographic Key Length Recommendation. `https://www.keylength.com/en/`, 2018. [Online; accessed April-2019].

[38] GrowthEnabler. Market pulse report, Internet of Things (IoT). Tech. rep., GrowthEnabler, 2017. [Online; accessed October-2019].

[39] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems 29*, 7 (Sept. 2013), 1645–1660.

[40] Gupta, A., Tripathi, M., Shaikh, T. J., and Sharma, A. A lightweight anonymous user authentication and key establishment scheme for wearable devices. *Computer Networks 149* (feb 2019), 29–42.

[41] Hankerson, D., Vanstone, S., and Menezes, A. *Guide to Elliptic Curve Cryptography.* Springer Professional Computing. Springer-Verlag New York, 2004.

[42] Hedabou, M., Pinel, P., and Bénéteau, L. A comb method to render ECC resistant against Side Channel Attacks. *IACR Cryptology ePrint Archive* (2004), 1–11.

[43] Heile, R. F., et al. Ieee standard for low-rate wireless networks. Tech. rep., Institute of Electrical and Electronics Engineers (IEEE), Apr. 2016.

[44] Hermans, J., Peeters, R., and Preneel, B. Proper rfid privacy: model and protocols. *IEEE Transactions on Mobile Computing 13*, 12 (2014), 2888–2902.

[45] Huang, J., Seberry, J., Susilo, W., and Bunder, M. Security analysis of michael: The ieee 802.11i message integrity code. In *Embedded and Ubiquitous Computing – EUC 2005 Workshops* (Berlin, Heidelberg, 2005), T. Enokido, L. Yan, B. Xiao, D. Kim, Y. Dai, and L. T. Yang, Eds., Springer Berlin Heidelberg, pp. 423–432.

[46] Hummen, R., Shafagh, H., Raza, S., Voig, T., and Wehrle, K. Delegation-based authentication and authorization for the IP-based Internet of Things. In *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)* (June 2014), IEEE, pp. 284–292.

[47] Hummen, R., Ziegeldorf, J. H., Shafagh, H., Raza, S., and Wehrle, K. Towards viable certificate-based authentication for the internet of things. In *Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy - HotWiSec '13* (New York, New York, USA, 2013), ACM Press, pp. 37–42.

[48] Iwata, T., Song, J., Lee, J., and Poovendran, R. The AES-CMAC Algorithm. RFC 4493, June 2006.

[49] Javdani, H., and Kashanian, H. Internet of things in medical applications with a service-oriented and security approach: a survey. *Health and Technology 8*, 1-2 (2018), 39–50.

[50] Kambourakis, G., Klaoudatou, E., and Gritzalis, S. Securing Medical Sensor Environments: The CodeBlue Framework Case. In *The Second International Conference on Availability, Reliability and Security (ARES'07)* (Apr. 2007), IEEE, pp. 637–643.

[51] Ko, J., Dutton, R. P., Lim, J. H., Chen, Y., Musvaloiu-E, R., Terzis, A., Masson, G. M., Gao, T., Destler, W., and Selavo, L. MEDiSN: Medical Emergency Detection in Sensor Networks. *ACM Transactions on Embedded Computing Systems 10*, 1 (Aug. 2010), 1–29.

[52] Koblitz, N. Elliptic curve cryptosystems. *Mathematics of computation 48*, 177 (1987), 203–209.

[53] Kocher, P., Jaffe, J., and Jun, B. Differential power analysis. In *Annual International Cryptology Conference* (1999), Springer, pp. 388–397.

[54] Kocher, P. C. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference* (1996), Springer, pp. 104–113.

[55] Kovatsch, M., Duquennoy, S., and Dunkels, A. Erbium (Er) REST Engine and CoAP Implementation for Contiki. `http://people.inf.ethz.ch/mkovatsc/erbium.php`, 2012. [Online; accessed 2017].

[56] Krawczyk, D. H., and Eronen, P. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869, May 2010.

[57] Kumar, P., Braeken, A., Gurtov, A., Iinatti, J., and Ha, P. H. Anonymous Secure Framework in Connected Smart Home Environments. *IEEE Transactions on Information Forensics and Security 12*, 4 (apr 2017), 968–979.

[58] Ledwaba, L. P. I., Hancke, G. P., Venter, H. S., and Isaac, S. J. Performance Costs of Software Cryptography in Securing New-Generation Internet of Energy Endpoint Devices. *IEEE Access 6* (2018), 9303–9323.

[59] Lee, J.-S., Su, Y.-W., and Shen, C.-C. A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. In *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society* (2007), IEEE, pp. 46–51.

[60] Lee, P. J., Lee, E. J., and Kim, Y. D. How to Implement Cost-Effective and Secure Public Key Cryptosystems. In *Cryptographic Hardware and Embedded Systems (CHES)* (Berlin, Heidelberg, 1999), Ç. K. Koç and C. Paar, Eds., Springer Berlin Heidelberg, pp. 73–79.

[61] Lee, Y. K., Batina, L., Singelée, D., and Verbauwhede, I. Low-cost untraceable authentication protocols for rfid. In *Proceedings of the Third ACM Conference on Wireless Network Security* (New York, NY, USA, 2010), WiSec '10, Association for Computing Machinery, p. 55–64.

[62] Libelium. Libelium Smart World Infographic – Sensors for Smart Cities, Internet of Things and beyond. `http://www.libelium.com/libelium-smart-world-infographic-smart-cities-internet-of-things/`, 2013. [Online; accessed November-2019].

[63] Liu, A., and Ning, P. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)* (St. Louis, MO, USA, Apr. 2008), IEEE, pp. 245–256.

[64] LoRa Alliance. Full end-to-end encryption for iot application providers. *A white paper prepared for the LoRa Alliance* (2017).

[65] Malan, D., Fulford-Jones, T., Welsh, M., and Moulton, S. CodeBlue : An Ad Hoc Sensor Network Infrastructure for Emergency

Medical Care. In *International Workshop on Wearable and Implantable Body Sensor Networks* (2004), WIBSN'04, pp. 12–14.

[66] MAXIM INTEGRATED. MAXREFDES100#: Health Sensor Platform. `https://www.maximintegrated.com/en/design/reference-design-center/system-board/6312.html`, 2019. [Online; accessed June-2019].

[67] MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. *Handbook of Applied Cryptography*, vol. 5. CRC Press, 2001.

[68] MILLER, V. S. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques* (1985), Springer, pp. 417–426.

[69] MONTENEGRO, G., SCHUMACHER, C., AND KUSHALNAGAR, N. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, 2007.

[70] MOOSAVI, S. R., GIA, T. N., NIGUSSIE, E., RAHMANI, A. M., VIRTANEN, S., TENHUNEN, H., AND ISOAHO, J. End-to-end security scheme for mobility enabled healthcare Internet of Things. *Future Generation Computer Systems 64* (2016), 108–124.

[71] MULLIGAN, G. The 6LoWPAN architecture. In *Proceedings of the 4th workshop on Embedded networked sensors - EmNets '07* (New York, New York, USA, 2007), ACM Press, pp. 1–78.

[72] NAVAS, R. E., LAGOS, M., TOUTAIN, L., AND VIJAYASANKAR, K. Nonce-based authenticated key establishment over OAuth 2.0 IoT proof-of-possession architecture. In *2016 IEEE 3rd World Forum Internet Things* (Dec. 2016), IEEE, pp. 317–322.

[73] NEUMAN, D. C., HARTMAN, S., RAEBURN, K., AND YU, T. The Kerberos Network Authentication Service (V5). RFC 4120, July 2005.

[74] NIGEL P. SMART. Algorithms, Key Size and Protocols Report (2018). Tech. rep., ECRYPT-CSA, Feb. 2018.

[75] NIST. Advanced encryption standard (AES). Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, Nov. 2001.

[76] OPENFOG CONSORTIUM ARCHITECTURE WORKING GROUP. OpenFog Reference Architecture for Fog Computing. Tech. rep., OpenFog Consortium, 2017.

[77] Padgette, J., Bahr, J., Batra, M., Holtmann, M., Smithbey, R., Chen, L., and Scarfone, K. Guide to bluetooth security. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, May 2017.

[78] Parliament, E. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *Official Journal of the European Union 2016*, 679 (2016), 88.

[79] Peeters, R., Hermans, J., and Fan, J. IBIHOP: Proper Privacy Preserving Mutual RFID Authentication. In *Workshop on RFID and IoT Security - RFIDSec Asia 2013* (Guangzhou,China, 2013), Cryptology and Information Security, IOS PRESS, pp. 45–56.

[80] Pfitzmann, A., and Hansen, M. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. *http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf 34* (01 2010), 98.

[81] Picek, S., Yang, B., Rozic, V., Vliegen, J., Winderickx, J., De Cnudde, T., and Mentens, N. Prngs for masking applications and their mapping to evolvable hardware. In *International Conference on Smart Card Research and Advanced Applications (CARDIS)* (Cannes, France, 2017), K. Lemke-Rust and M. Tunstall, Eds., Springer International Publishing, pp. 209–227.

[82] Pierrard, A., and Dumoulin, J. wearIT4health: Wearable Integrated Technology for health monitoring of hospitalised patients in the Euregio Meuse-Rhine. http://www.wearit4health.com/, 2019. [Online; accessed December-2019].

[83] Pivotal Software Incorporated. Rabbitmq is the most widely deployed open source message broker. `https://www.rabbitmq.com/`, 2019. [Online; accessed March-2019].

[84] Rabbani, M. M., Vliegen, J., Winderickx, J., Conti, M., and Mentens, N. SHeLA: Scalable Heterogeneous Layered Attestation. *IEEE Internet of Things Journal* (2019), 12 pages.

[85] Raza, S., Seitz, L., Sitenkov, D., and Selander, G. S3K: Scalable Security with Symmetric Keys - DTLS Key Establishment for the Internet of Things. *IEEE Transactions on Automation Science and Engineering 13*, 3 (2016), 1270–1280.

[86] Rescorla, E. RFC8446: The Transport Layer Security (TLS) Protocol Version 1.3. Tech. rep., IETF, 2018.

[87] Rescorla, E., and Dierks, T. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Aug. 2008.

[88] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (1978), 120–126.

[89] Roussey, C., Bernard, S., Andre, G., Corcho, O., Sousa, G. D., Boffety, D., and Chanet, J.-P. Short Paper: Weather Station Data Publication at IRSTEA: An Implementation Report. In *Joint Proceedings of the 6th International Workshop on the Foundations, Technologies and Applications of the Geospatial Web and 7th International Workshop on Semantic Sensor Networks (TC-SSN)* (Aachen, 2014), K. Kyzirakos, R. Gruetter, D. Kolas, M. Perry, M. Compton, K. Janowicz, and K. Taylor, Eds., no. 1401 in CEUR Workshop Proceedings, pp. 89–104.

[90] Rubin, A., and Honeyman, P. Nonmonotonic cryptographic protocols. In *Proceedings The Computer Security Foundations Workshop VII* (June 1994), IEEE Comput. Soc. Press, pp. 100–116.

[91] Saeed, M. E. S., Liu, Q.-Y., Tian, G., Gao, B., and Li, F. AKAIoTs: authenticated key agreement for Internet of Things. *Wireless Networks 25*, 6 (Aug. 2019), 3081–3101.

[92] Saied, Y. B., and Olivereau, A. D-HIP: A distributed key exchange scheme for HIP-based Internet of Things. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (June 2012), IEEE, pp. 1–7.

[93] Saied, Y. B., Olivereau, A., Zeghlache, D., and Laurent, M. Lightweight collaborative key establishment scheme for the Internet of Things. *Computer Networks 64* (May 2014), 273–295.

[94] Samie, F., Bauer, L., and Henkel, J. An approximate compressor for wearable biomedical healthcare monitoring systems. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)* (Oct. 2015), IEEE, pp. 133–142.

[95] Schaad, J. RFC 8152: CBOR Object Signing and Encryption (COSE). Tech. rep., Internet Engineering Task Force (IETF), July 2017.

[96] Schnorr, C.-P. Efficient identification and signatures for smart cards. In *Advances in Cryptology – Proceedings of CRYPTO* (1989), Lecture Notes in Computer Science, Springer-Verlag, pp. 239–252.

[97] Selander, G., Mattsson, J., Palombini, F., and Seitz, L. Object Security for Constrained RESTful Environments (OSCORE). Tech. rep., Internet Engineering Task Force (IETF), July 2019.

[98] Semtech Corporation. SX1272/73 - 860 MHz to 1020 MHz Low Power Long Range Transceiver. `https://www.semtech.com/uploads/documents/SX1272_DS_V4.pdf`, 2019. [Online; accessed June-2019].

[99] Sen, S., Koo, J., and Bagchi, S. TRIFECTA: Security, Energy Efficiency, and Communication Capacity Comparison for Wireless IoT Devices. *IEEE Internet Computing 22*, 1 (Jan. 2018), 74–81.

[100] Shelby, Z., Hartke, K., and Bormann, C. RFC 7252: The Constrained Application Protocol (CoAP). Tech. rep., Internet Engineering Task Force (IETF), June 2014.

[101] Shostack, A. *Threat Modeling: Designing for Security.* Wiley, 2014.

[102] Shuai, M., Liu, B., Yu, N., and Xiong, L. Lightweight and Secure Three-Factor Authentication Scheme for Remote Patient Monitoring Using On-Body Wireless Networks. *Security and Communication Networks 2019* (jun 2019), 1–14.

[103] Smart, N. P., Rijmen, V., Gierlichs, B., Paterson, K., Stam, M., Warinschi, B., Watson, G., and Tirtea, R. Algorithms key sizes and parameters report-2014. *European Union Agency for Network and Information Security (ENISA) TP-05-14-084-EN-N* (2014), 113.

[104] Sornin, N., Luis, M., Eirich, T., Kramp, T., and Hersent, O. LoRaWAN Specification V1.0. Tech. rep., LoRa Alliance, 2015.

[105] STMicroelectronics. STM32 Nucleo-64 development board with STM32L073RZ MCU, supports Arduino and ST morpho connectivity. `https://www.st.com/en/evaluation-tools/nucleo-l073rz.html`, 2019. [Online; accessed June-2019].

[106] Technical Committee, I. J. . ISO/IEC 7498-1:1994: Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model. Tech. rep., International Organization for Standardization (ISO), 1994.

[107] technology, I. J. . I. Information technology – Message Queuing Telemetry Transport (MQTT) v3.1.1. ISO/IEC 20922:2016, International Organization for Standardization (ISO), June 2016.

[108] Texas Instruments. CC3120 SimpleLink™ Wi-Fi® Wireless Network Processor, Internet-of-Things Solution for MCU Applications. `http://www.ti.com/lit/ds/swas034/swas034.pdf`, 2017. [Online; accessed June-2019].

[109] Texas Instruments Incorporated. Msp432p4011: Simplelink ultra-low-power 32-bit arm cortex-m4f mcu with precision adc, 2mb flash and 256kb ram. `http://www.ti.com/product/MSP432P4011`, 2019. [Online; accessed March-2019].

[110] Texas Instruments Incorporated. Simplelink™ msp432p401r high-precision adc launchpad™ development kit. `http://www.ti.com/tool/MSP-EXP432P401R`, 2019. [Online; accessed June-2019].

[111] The BIPM and the Metre Convention. The International System of Units (SI). Tech. rep., Bureau International des Poids et Mesures (BIPM), 2019.

[112] The Things Network. Duty Cycle for LoRaWAN Devices. `https://www.thethingsnetwork.org/docs/lorawan/duty-cycle.html`. [Online; accessed June-2019].

[113] Trappe, W., Howard, R., and Moore, R. S. Low-Energy Security: Limits and Opportunities in the Internet of Things. *IEEE Security & Privacy 13*, 1 (Jan. 2015), 14–21.

[114] Tschofenig, H., and Eronen, P. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279, Dec. 2005.

[115] Van Den Abeele, F., Hoebeke, J., Moerman, I., and Demeester, P. Integration of Heterogeneous Devices and Communication Models via the Cloud in the Constrained Internet of Things. *International Journal of Distributed Sensor Networks 2015* (2015).

[116] Whiting, D., Housley, R., and Ferguson, N. Counter with CBC-MAC (CCM). Tech. rep., Internet Engineering Task Force (IETF), Sept. 2003.

[117] Wi-Fi Alliance. Who We Are. `https://www.wi-fi.org/who-we-are`, 2019. [Online; accessed October-2019].

[118] Winderickx, J., Bellier, P., Coppieters, D., Duflot, P., and Mentens, N. Communication and security trade-offs for battery-powered devices: a case study on wearable medical sensor systems. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* (2019), 17 pages. [submitted].

[119] Winderickx, J., Bellier, P., Duflot, P., Coppieters, D., and Mentens, N. Work-in-Progress: Communication and security trade-offs for wearable medical sensor systems in hospitals. In *Proceedings of the International Conference on Embedded Software Companion - EMSOFT '19* (New York, New York, USA, 2019), S. Sankaranarayanan and T. Bourke, Eds., ACM Press, pp. 1–2.

[120] Winderickx, J., Braeken, A., and Mentens, N. Storage and computation optimization of public-key schemes on embedded devices. In *2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech)* (Brussels, Belgium, 2018), IEEE, pp. 1–8.

[121] Winderickx, J., Braeken, A., and Mentens, N. Enhanced end-to-end security through symmetric-key cryptography in wearable medical sensor networks. *ACM Transactions on Computing for Healthcare (HEALTH)* (2019), 17 pages. [submitted].

[122] Winderickx, J., Braeken, A., Singelée, D., and Mentens, N. In-depth energy analysis of security algorithms and protocols for the Internet of Things. *ACM Transactions on Internet of Things (TIOT)* (2019), 18 pages. [submitted].

[123] Winderickx, J., Braeken, A., Singelée, D., Peeters, R., Vandenryt, T., Thoelen, R., and Mentens, N. Digital signatures and signcryption schemes on embedded devices. In *Proceedings of the 15th ACM International Conference on Computing Frontiers - CF '18* (Ischia, Italy, 2018), M. Moretó and J. Weidendorfer, Eds., ACM Press, pp. 342–347.

[124] Winderickx, J., Daemen, J., and Mentens, N. Exploring the use of shift register lookup tables for Keccak implementations on Xilinx FPGAs. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)* (Lausanne, Switzerland, 2016), J. Anderson and P. Brisk, Eds., IEEE, pp. 454–457.

[125] Winderickx, J., Daemen, J., and Mentens, N. On the parallelization of slice-based Keccak implementations on Xilinx FPGAs. In *6th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016)* (Barcelona, Spain, 2016), G. D. Natale and I. Polian, Eds., pp. 103–107.

[126] Winderickx, J., Singelée, D., and Mentens, N. HeComm: End-to-end secured communication in a heterogeneous IoT environment via fog computing. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (Las Vegas, Nevada, USA, 2018), B. Lee and J. Kang, Eds., IEEE, pp. 1–6.

[127] WolfSSL. wolfCrypt Embedded Crypto Engine. https://www.wolfssl.com/products/wolfcrypt-2/, 2019. [Online; accessed June-2019].

[128] Wuyts, K., and Joosen, W. Linddun privacy threat modeling: a tutorial, 2015.

[129] Xu, L. D., He, W., and Li, S. Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics 10*, 4 (Nov. 2014), 2233–2243.

[130] Xu, Y., and Xie, X. Analysis of Authentication Protocols Based on Rubin Logic. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing* (oct 2008), IEEE, pp. 1–5.

[131] Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. Internet of Things for Smart Cities. *IEEE Internet of Things Journal 1*, 1 (Feb. 2014), 22–32.

[132] Zhang, Z.-K., Cho, M. C. Y., Wang, C.-W., Hsu, C.-W., Chen, C.-K., and Shieh, S. IoT Security: Ongoing Challenges and Research Opportunities. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications* (Nov. 2014), IEEE, pp. 230–234.

[133] Zhao, K., and Ge, L. A Survey on the Internet of Things Security. In *2013 Ninth International Conference on Computational Intelligence and Security* (Dec. 2013), IEEE, pp. 663–667.

[134] Zolertia. Zolertia Z1 mote. https://github.com/Zolertia/Resources/wiki/The-Z1-mote, 2016. [Online; accessed 2017].

List of publications

International journal submissions

2019 WINDERICKX, J., BELLIER, P., COPPIETERS, D., DUFLOT, P., AND MENTENS, N. Communication and security trade-offs for battery-powered devices: a case study on wearable medical sensor systems. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* (2019), 17 pages. [submitted]

2019 WINDERICKX, J., BRAEKEN, A., SINGELÉE, D., AND MENTENS, N. In-depth energy analysis of security algorithms and protocols for the Internet of Things. *ACM Transactions on Internet of Things (TIOT)* (2019), 18 pages. [submitted]

2019 WINDERICKX, J., BRAEKEN, A., AND MENTENS, N. Enhanced end-to-end security through symmetric-key cryptography in wearable medical sensor networks. *ACM Transactions on Computing for Healthcare (HEALTH)* (2019), 17 pages. [submitted]

International journal papers

2019 RABBANI, M. M., VLIEGEN, J., WINDERICKX, J., CONTI, M., AND MENTENS, N. SHeLA: Scalable Heterogeneous Layered Attestation. *IEEE Internet of Things Journal* (2019), 12 pages

International conference papers

2016 WINDERICKX, J., DAEMEN, J., AND MENTENS, N. Exploring the use of shift register lookup tables for Keccak implementations on Xilinx FPGAs. In *2016 26th International Conference on Field Programmable Logic and*

Applications (FPL) (Lausanne, Switzerland, 2016), J. Anderson and P. Brisk, Eds., IEEE, pp. 454–457

2016 Winderickx, J., Daemen, J., and Mentens, N. On the parallelization of slice-based Keccak implementations on Xilinx FPGAs. In *6th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016)* (Barcelona, Spain, 2016), G. D. Natale and I. Polian, Eds., pp. 103–107

2017 Picek, S., Yang, B., Rozic, V., Vliegen, J., Winderickx, J., De Cnudde, T., and Mentens, N. Prngs for masking applications and their mapping to evolvable hardware. In *International Conference on Smart Card Research and Advanced Applications (CARDIS)* (Cannes, France, 2017), K. Lemke-Rust and M. Tunstall, Eds., Springer International Publishing, pp. 209–227

2018 Winderickx, J., Singelée, D., and Mentens, N. HeComm: End-to-end secured communication in a heterogeneous IoT environment via fog computing. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (Las Vegas, Nevada, USA, 2018), B. Lee and J. Kang, Eds., IEEE, pp. 1–6

2018 Winderickx, J., Braeken, A., Singelée, D., Peeters, R., Vandenryt, T., Thoelen, R., and Mentens, N. Digital signatures and signcryption schemes on embedded devices. In *Proceedings of the 15th ACM International Conference on Computing Frontiers - CF '18* (Ischia, Italy, 2018), M. Moretó and J. Weidendorfer, Eds., ACM Press, pp. 342–347

2018 Winderickx, J., Braeken, A., and Mentens, N. Storage and computation optimization of public-key schemes on embedded devices. In *2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech)* (Brussels, Belgium, 2018), IEEE, pp. 1–8

2019 Winderickx, J., Bellier, P., Duflot, P., Coppieters, D., and Mentens, N. Work-in-Progress: Communication and security trade-offs for wearable medical sensor systems in hospitals. In *Proceedings of the International Conference on Embedded Software Companion - EMSOFT '19* (New York, New York, USA, 2019), S. Sankaranarayanan and T. Bourke, Eds., ACM Press, pp. 1–2

FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING
ES&S AND IMEC-COSIC
Wetenschapspark 27
B-3590 Diepenbeek
nele.mentens@kuleuven.be
https://iiw.kuleuven.be/onderzoek/ess