DOCTORAL THESIS

Secure and Reliable Communication Protocols in Industrial IoT Networks

by

Antonio Javier Cabrera Gutierrez

Directed by

Luis Parrilla Roure

and

Encarnación Castillo Morales

in the

University of Granada

Department of Electronics and Computer Technology

under the

Doctoral Programme in Information and Communication Technologies

developed in

Infineon Technologies AG

October 2023

DOCTORAL THESIS

Secure and Reliable Communication Protocols in Industrial IoT Networks

Directed by:

Prof. Luis Parrilla Roure

Prof. Encarnación Castillo Morales

Developed at:

Infineon Technologies AG

BEX RDE RDF EET

Munich, Germany

Editor: Universidad de Granada. Tesis Doctorales
Autor: Antonio Javier Cabrera Gutiérrez
ISBN: 978-84-1195-174-6
URI: https://hdl.handle.net/10481/89419

Abstract

Faculty of Science

Department of Electronics and Computer Technology

Secure and Reliable Communication Protocols in Industrial IoT Networks

by Antonio Javier Cabrera Gutierrez

The security and reliability of network protocols have undergone a major transformation in recent years, and new network architectures such as industrial Internet of Things (IoT) networks have led to the emergence of new communication paradigms such as decentralised Blockchain networks. These networks represent a substantial advance in terms of system security, but without neglecting the physical security of IoT devices, which must be protected with hardware security elements. The joint application of Blockchain technologies with hardware security is essential in these new architectures.

This doctoral thesis is the result of the research carried out on these technologies and their application in industrial IoT environments. Thus, this work describes the integration of Blockchain technologies and hardware security devices, pointing out which protocols and requirements are necessary to carry out a successful integration. Subsequently, the application of Blockchain and hardware security elements in virtualised environments has been investigated by applying them to microservices-based architectures. The use of Blockchain in industrial IoT networks introduces new concepts such as Oracles, which are entities that provide truthful information to the Blockchain network. This concept is discussed in this thesis, where a design is proposed using a two-core hardware platform isolating one of them for the execution of a secure environment. The prototype of this concept is using in a logistic use case for wine supply chain. Finally, a use case is presented, by simulating a smart grid where different entities exchange energy certificates without the intervention of third parties, thus automating processes through smart contracts.

The research has been carried out under a doctoral contract at the premises of Infineon Technologies AG, at its headquarters in Munich, Germany.

Acknowledgements

At the moment of writing these words, one finds oneself at the end of the road that has led to this moment: the culmination of a doctoral thesis of some four years in which, with no little difficulty, the established objectives have been achieved, ad Augusta, per Angusta. Only at this precise moment and looking back, my eyes - the distance - allow me to contemplate the road travelled, and although I can make out the odd pothole and the odd pebble on it, the simple fact that the path I have traced has reached the established destination fills me with immense happiness.

This path has not been travelled only by the writer of these lines, in which case, if it had been, I would certainly not have been able to reach this point, not even a small part of the way! That is why I would like to thank all the people who have accompanied me on this journey, without whom none of what is set out in black and white in this thesis would have been possible.

First of all, I would like to thank my tutors, Encarni and Luis, without whom it would not have been possible to reach this point or even begin the journey. Above all, for all their infinite effort and dedication in the corrections made to the publications, without downplaying the advice and help they have given me over the four years in choosing journals and the suggestions for changes that the reviewers requested.

To Diego and Antonio, the two main promoters of having started this path. It seems like only yesterday when one afternoon in March 2017, in an Electronic Systems Design class, Professor Diego told us about Infineon and the possibility of doing an internship during the summer. Even today, I still can't explain how a person like me, who had hardly travelled or left his homeland, would venture a few months later to take the decision to go and live in another country. Perhaps it was to avoid the bad taste in my mouth of not having taken the risk and been brave, what I do know is that what pushed me to take the decision was the fear of not taking it, as Bergamín said: "Courage waits, fear goes looking". There I met many people who welcomed me and made me feel at home. Among them Antonio, who started out as my supervisor and today, although he still is, I consider that he has also become (I rarely say these things) a great friend. Thank you for all the help you have given me since I started working at Infineon in 2017, for your advice and your wisdom, without them I would not have got to where I am now.

Nor can I forget those people who have accompanied me during this journey in Germany: Juan Cruz, Jesus, Javi, Jakob, Miguel, Angel, Adrián, Juan Fernandez, Borja and Fernando. And, in spite of the distance, my great memory to Pedro and Isaac, whose laughs, anecdotes and conversations by Telegram have made me feel as if I had

not left Spain. As well as to my friends from Noalejo, who are always in my memory despite the distance: Juanfran, David, Samuel, etc. Thank you.

And, finally, to the most important people, my family, without whose effort and sacrifice during so many years I would not have made it this far. No matter how many words of thanks I write to you below, they would not be enough to express all my gratitude, so I will be brief: Thank you very much from the bottom of my heart for everything you have done for me. This thesis is your fruit.

Resumen

Facultad de Ciencias
Departamento de Electronica y Tecnologia de Computadores

Protocolos de Comunicaciones Seguros y Fiables en Redes Industriales de IoT

por Antonio Javier Cabrera Gutierrez

La seguridad y fiabilidad en los protocolos de red han sufrido una gran transformación en los últimos años de la mano de nuevas arquitecturas de red como las redes industriales de dispositivos del Internet de las Cosas (Internet of Things, IoT), que han hecho aparecer nuevos paradigmas de comunicaciones como las redes descentralizadas de Blockchain. Estas redes suponen un avance sustancial en cuanto a la seguridad de los sistemas se refiere, pero sin descuidar a la seguridad física de los dispositivos IoT, los cuales, han de ser protegidos con elementos de seguridad hardware. Así, la aplicación conjunta de tecnologías de Blockchain con la seguridad hardware se muestra imprescindible en estas nuevas arquitecturas. Esta tesis doctoral es el resultado de la investigación realizada sobre estas tecnologías y su aplicación en entornos industriales de IoT. En la misma se describe la integración de las tecnologías de Blockchain y de dispositivos de seguridad hardware, señalando qué protocolos y qué requisitos son los necesarios para llevar a cabo una integración exitosa. Seguidamente, se ha investigado la aplicación de Blockchain y de los elementos de seguridad hardware en entornos virtualizados, aplicándolos a las arquitecturas basadas en microservicios. El uso de Blockchain en redes industriales de IoT introduce nuevos conceptos como son los Oráculos, estas entidades proveen información veraz a la red de Blockchain. Este concepto se discute en esta tesis, donde se propone un diseño usando una plataforma hardware de dos núcleos aislando uno de ellos para la ejecución de un entorno seguro. El prototipo se demuestra en un caso de uso relacionado con la cadena logistica del vino. Finalmente, se expone un caso de uso, simulando una "*smart grid*" donde diferentes entidades intercambian certificados de energia verde sin la intervención de terceros, automatizando procesos a través de "*smart contracts*".

La investigación se ha realizado bajo un contrato doctoral en las instalaciones de Infineon Technologies AG, en su sede principal de Múnich, Alemania.

Agradecimientos

En el momento de escribir estas palabras, uno se encuentra al final del camino que ha supuesto el llegar a este momento: el culminar una tesis doctoral de unos cuatro años en los que, no con pocas dificultades, se han ido cumpliendo los objetivos establecidos, *ad Augusta, per Angusta.* Solo en este preciso instante y volviendo la vista atrás, mis ojos —la distancia— me permiten contemplar el camino andado, y aunque puedo divisar algún bache y alguna que otra piedrecita en él, el simple hecho de que ese camino trazado ha llegado al destino establecido, me llena de inmensa felicidad.

Ese camino trazado no ha sido recorrido solamente por el que escribe estas líneas, en cuyo caso, de ser así, con toda certeza no hubiera sido capaz de llegar a este punto, ¡ni tan siquiera a una mínima parte del recorrido! Es por esto, por lo que quisiera dar las gracias a todas las personas que me han acompañado durante este trayecto, sin las cuales, no hubiera sido posible nada de lo que en esta tesis se plasma en negro sobre blanco.

En primer lugar, me gustaría dar un profundo agradecimiento a mis tutores, Encarni y Luis, sin los cuales no hubiera sido posible llegar hasta aquí ni tan siquiera empezar el camino. Sobre todo, por todo su infinito esfuerzo y dedicación en las correcciones hechas a las publicaciones, sin quitar importancia a los consejos y ayuda que me han dado durante los cuatro años a la hora de elegir revistas y las sugerencias sobre los cambios que los revisores pedían.

A Diego y Antonio, los dos promotores principales de haber iniciado este camino. Parece que fue ayer cuando una tarde de marzo de 2017 en clase de Diseño de Sistemas Electrónicos, el profesor Diego nos habló de Infineon y de la posibilidad de realizar un *internship* durante el verano. Aun a día de hoy, no me explico cómo una persona como yo, que apenas había viajado ni salido de su patria chica, se aventuraría unos meses más tardes a tomar la decisión de irse a vivir a otro país. Quizás sea para no quedarse uno con el mal sabor de boca de no haber arriesgado y sido valiente, lo que si se es que lo que me empujo a tomar la decisión fue el miedo de no tomarla, como dijo Bergamín: *"El valor espera, el miedo va a buscar"*. Allí conocí a muchas personas que me acogieron haciéndome sentir como en casa. Entre ellas Antonio, que empezó siendo mi supervisor y a día de hoy, aunque todavía sigue siéndolo, considero que también se ha convertido en (pocas veces suelo decir estas cosas) un gran amigo. Gracias por toda la ayuda prestada desde que empecé trabajando en Infineon en 2017, por tus consejos y tu sabiduría, sin ellos no hubiera llegado a donde estoy ahora.

Tampoco me puedo olvidar de esas personas que me han acompañado durante este periplo en Alemania: A Juan Cruz, Jesus, Javi, Jakob, Miguel, Angel, Adrián, Juan

Fernandez y Fernando. Y, a pesar de la distancia, mi recuerdo enorme a Pedro e Isaac, cuyas risas, anécdotas y conversaciones por Telegram han hecho sentirme como si no me hubiera ido de España. Así como también a mis amigos de Noalejo, en los que siempre están en mi recuerdo y memoria a pesar de la distancia: Juanfran, David, Samuel, etc. Gracias.

Y, finalmente, a las personas más importantes, mi familia, sin cuyo esfuerzo y sacrificio durante tantos años no me hubieran hecho llegar hasta aquí. Por muchas palabras de agradecimiento que escribiera a continuación hacia vosotros, no serían suficientes para expresar toda mi gratitud, así que seré breve: Muchas gracias de todo corazón por todo lo que habéis hecho por mí. Esta tesis es fruto vuestro.

A mis padres: Antonio y Francisca

"El trabajo que nunca se empieza es el que tarda más en finalizarse."

J. R. R. Tolkien

Contents

Abstract iii

Acknowledgements v

Resumen vii

Agradecimientos ix

List of Figures xix

List of Tables xxi

Abbreviations xxiii

I Introduction 1

1 Introduction 3
1.1 Objectives . . . 4
1.2 Motivation . . . 4

2 Outline/Thesis Structure 7

3 Funding and Methodology 9
3.1 Funding projects related to the Thesis . . . 9
3.2 Methodology . . . 12

II Publications 15

4 Integration of Hardware Security Modules and Permissioned Blockchain in Industrial IoT Networks 17
4.1 Introduction . . . 18
4.2 State of the art . . . 21
4.3 Background . . . 24
4.4 Integration between Hyperledger Fabric and Trusted platform module . . . 30
4.5 Conclusion . . . 43

4.6 Acknowledgements . . . 44
4.7 References . . . 44

5 Blockchain-Based Services Implemented in a Microservices Architecture Using a Trusted Platform Module Applied to Electric Vehicle Charging Stations 51
5.1 Introduction . . . 52
5.2 Related Work . . . 55
5.3 Trusted Platform Modules in Combination with Micro-services-Based Architectures . . . 59
5.4 Architecture Design . . . 62
5.5 Electrical Vehicles Charging Stations Use Case . . . 73
5.6 Conclusions . . . 77
5.7 Acknowledgements . . . 78
5.8 References . . . 79

6 Secure Sensor Prototype Using Hardware Security Modules and Trusted Execution Environments in a Block-chain Application: Wine Logistic Use Case 87
6.1 Introduction . . . 88
6.2 Related Works . . . 91
6.3 Design Proposed . . . 92
6.4 Blockchain Application . . . 100
6.5 Wine Logistic Use Case . . . 104
6.6 Conclusions . . . 109
6.7 Acknowledgement . . . 109
6.8 References . . . 110

III Conference publications 117

7 Integration of Hardware Security Modules in Blockchain Networks 119
7.1 Introduction . . . 120
7.2 General description . . . 121
7.3 Conclusions . . . 123
7.4 Acknowledgments . . . 124
7.5 References . . . 124

8 Blockchain-based implementation of Tradable Green Certificates 127
8.1 Introduction . . . 128
8.2 State of the art . . . 129
8.3 Architecture design . . . 129
8.4 Implementation . . . 132
8.5 Conclusions . . . 135
8.6 Acknowledgements . . . 135
8.7 References . . . 135

IV Conclusions 139

9 Conclusions 141

List of Figures

4.1 Typical PKI infrastructure vs. decentralized ledger infrastructure. 22
4.2 TPM2.0 architecture. 26
4.3 Enrollment mechanism. 33
4.4 Signature procedure mechanism between the TPM2.0 and the Peer node. 34
4.5 Proof of concept, Raspberry Pi 4 Model B with Hyperledger Fabric. . . . 35
4.6 Admin and User enrollment process. 36
4.7 Transaction mechanism between Client node and the Peer nodes. 37
4.8 Comparison between TPM access time and IIoT data collection procedure. 40

5.1 Blockchain with HSM. 54
5.2 vTPM architecture. 56
5.3 Hypervisor-based virtualization vs container-based virtualization. 57
5.4 Example of microservice-based architecture. 58
5.5 Approach with vTPM inside each container. 60
5.6 Approach with vTPM in the OS kernel. 61
5.7 Approach with vTPM in a dedicated container. 61
5.8 TPM software. 64
5.9 vTPM architecture. 65
5.10 Container attestation mechanism. 68
5.11 Seal and unseal processes. 70
5.12 Authentication process with Cognito AWS. 72
5.13 EVs charging microservice-based architecture porposed. 74
5.14 User authentication process with Cognito AWS. 77

6.1 IIoT architecture. 89
6.2 Design proposed. 93
6.3 Secure sensor concept. 94
6.4 Trusted Firmware-M. 96
6.5 Driver implementation in TF-M. 98
6.6 Operations carried out by the HSM. 99
6.7 Timestamp mechanism. 104
6.8 PSoC64-based edge node. 105
6.9 Architecture proposed in wine logistic use case. 106
6.10 PSoC64 located in a barrel with the data records gathering by sensors. . . 106

7.1 Public Key Infrastructure 121
7.2 Decentralized PKI 122

8.1 HLF architecture. 131
8.2 Mechanisms where HSM is involved. 132

8.3 RPI with HSM 133
8.4 Hardware architecture demonstrator. 134
8.5 Interaction between producer and consumer 134

List of Tables

4.1 Execution time of the different approaches in milliseconds. 40
4.2 Security features of each approach. 41

6.1 Blockchain ledger example. 107
6.2 Performance evaluation in milliseconds comparing three approaches. . . . 108

8.1 Information keeping in the ledger related to the nodes. 133
8.2 Information related transaction of TGCs. 133

Abbreviations

ABE	**A**ttribute-**B**ased **E**ncryption
ABRMD	**A**ccess **B**roker and **R**esource **M**anagement **D**aemon
AES	**A**dvanced **E**ncryption **S**tandard
AIK	**A**ttestation **I**dentity **K**ey
API	**AP**plication **I**nterface
AWS	**A**mazon **W**eb **S**ervices
BLE	**B**luetooth **L**ow **E**nergy
BLOB	**B**inary **L**arge **OB**ject
CA	**C**ertification **A**uthority
CI/CD	**C**continuous **I**ntegration **C**continuous **D**evelopment
CSR	**C**ertificate **S**igning **R**equest
DI	**D**igital **I**dentity
DLT	**D**ecentralized **L**edger **T**echnologies
DoS	**D**enial **o**f **S**ervice
DPoS	**D**elegated **P**roof **o**f **S**take
ECC	**E**lliptic **C**ryptographic **C**urve
ECDSA	**E**lliptic **C**urve **D**igital **S**ignature **A**lgorithm
EK	**E**ndorsement **K**ey
ESAPI	**E**nhanced **S**ystem **A**PI
EV	**E**lectrical **V**ehicles
FAPI	**F**eature **A**PI
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
GPIO	**G**eneral **P**urpose **I**nput **O**utput
HLF	**H**yper**l**edger **F**abric

HSM	**H**ardware **S**ecurity **M**odule
HW	**H**ard**w**are
I2C	**I**nter-**I**ntegrated **C**ircuit
IaaS	**I**nfraestructure **a**s **a** **S**ervice
IIoT	**I**ndustrial **I**nternet **o**f **T**hings
IoT	**I**nterent **o**f **T**hings
MMIO	**M**emory **M**apped **I**nput **O**utput
MQTT	**M**essage **Q**ueue **T**elemetry **T**ransport
NFC	**N**arrow **F**ield **C**ommunication
NSPE	**N**on-**S**ecure **P**rocessing **E**nvironment
NTP	**N**etwork **T**ime **P**rotocol
NTS	**N**etwork **T**ime **S**ecurity
OS	**O**perating **S**ystem
OSSL ENG	**O**pen **SSL** **E**ngine
PaaS	**P**latform **a**s **a** **S**ervice
PB	**P**ermissioned **B**lockchain
PCR	**P**latform **C**onfiguration **R**egister
PKCS	**P**ublic **K**ey **C**ryptographic **S**tandard
PKI	**P**ublic **K**ey **I**nfraestructure
PoS	**P**roof **o**f **S**take
PoW	**P**roof **o**f **W**ork
PRNG	**P**seudo-**R**andom **N**umber **G**enerator
PSA	**P**latform **S**ecurity **A**rchitecture
PUF	**P**hysical **U**nclonable **F**unctions
REC	**R**enewable **E**nergy **C**ertificate
REST	**RE**presentational **S**tate **T**ransfer
RNG	**R**andom **N**umber **G**enerator
RoT	**R**oot **o**f **T**rust
RPI	**R**aspberry **PI**
RSA	**R**ivest **S**hamir **A**dleman
SaaS	**S**oftware **a**s **a** **S**ervice
SAPI	**S**ystem **A**PI
SPE	**S**ecure **P**rocessing **E**nvironment

SPI	**S**erial **P**eripheral **I**nterface
SPM	**S**ecure **P**artition **M**anager
SPOF	**S**ingle **P**oint **O**f **F**ailure
SW	**S**oft**w**are
TCG	**T**rusted **C**omputing **G**roup
TCTI	**T**PM **C**ommand **T**ransmission **I**nterface
TEE	**T**rusted **E**xecution **E**nvironment
TF-M	**T**rusted **F**irmware-**M**
TGC	**T**radable **G**reen **C**ertificate
TLS	**T**ransport **L**ayer **S**ecurity
TPM	**T**rusted **P**latform **M**odule
TRNG	**T**rue **R**andom **N**umber **G**enerator
TSS	**T**PM **S**oftware **S**tack
VM	**V**irtual **M**achine
vTPM	**v**irtual **T**PM

Part I

Introduction

"Bisogna fare la propria vita come si fa un'opera d'arte"

Gabrielle D'Annunzio

1

Introduction

THe evolution of information technologies in recent years has led of social, economic and cultural changes over the past years. The incorporation of new technologies into people's daily lives as well as into industry has brought about a profound change in the way we understand the paradigms of communication that have traditionally been used since the first revolution of the technological era.

Traditional communications networks and paradigms are now obsolete and have serious difficulties for applications in Industry4.0 environments [1]. At the present time, there are increasing problems facing these technologies in terms of security. For example, in the case of communications systems, identification and authentication has traditionally been realised using a Public Key Infrastructure following a client-server paradigm. In recent years, security breaches discovered in this type of paradigm have led to the development of new technologies to address such shortcomings [2].

In this sense, technologies such as Blockchain come to fill this kind of security gaps. Originally, this technology was applied to cryptocurrencies such as Bitcoin. Although in the beginning this technology was only used for this type of application, its field of application has recently become more relevant. Today there are applications in different fields as [3], business [4], economics [5], health [6], or energy [7]. The reason for this rise is mainly due to the ability of this technology to address security problems that traditional technologies did not address because they were not able to mitigate them or simply because the sophisticated ways of attacking a system available now, did not exist when they were devised.

Another technology that has also had a great impact in recent years is hardware security devices, especially with the rise of Blockchain. Indeed, these devices have been widely

used as hardware wallets for the storage of cryptographic keys required by Blockchain, due to the hardware-based root of trust generated by them, wich ensures certified protection agaist hardware attacks [8].

This doctoral thesis deals with the integration of this type of technology and its possible applications. In this way, a proposal for the integration of Blockchain networks and an Hardware Security Module (HSM) is discussed, as well as different applications of these technologies working together in energy exchanges within the energy market.

The document is structured as follows: the introductory chapter gathers a brief of the State of the Art, objectives and motivation. The following chapter, Chapter 2, covers the publications achieved. Chapter 3 deals with the methodology aspect and materials related to the development of the different results achieved during the thesis. The second part of the document is structured in chapters which each of them correspond with one publication described above. The third part is composed of articles presented in conferences. Finally, Chapter 8 summarize the results achieved in this doctoral period.

1.1 Objectives

The objectives of this doctoral thesis are as follow:

- To propose a proof of concept integrating a Blockchain-enabled HSM for an Internet of Things (IoT) node with the ability to perform traditional cryptographic operations.
- To study different applications where this type of concepts can be applied in the world of industry and to make an application for a real environment.
- To design a secure sensor that works as a Blockchain Oracle to reliably provide data. This secure sensor must use different technologies that provide security to the system whose main function is to provide a hardware solution for secure data extraction in Industrial IoT (IIoT) environments.
- To provide security mechanisms for a virtualised environment. Many IIoT architectures have cloud platforms that are vulnerable to attacks and on which different applications can perform cryptographic operations. In this case a virtualisable system based on an HSM is necessary.

1.2 Motivation

As mentioned above, in recent times we have experienced great progress in improving our lives due to the advent of new technologies. However, emerging technologies such

as the aforementioned Blockchain tend to have a very slow process of immersion in the industrial world. This is often due to different reasons, especially with technologies such as Blockchain, where there are always different opinions among the scientific community, which results in a sceptical view of the application of this type of technology in the industrial world.

There is usually a gap between scientific innovation and the industrial world which is always difficult to close and it is only after some time that emerging technologies achieve a certain degree of maturity.

The motivation of this doctoral thesis is to try to close this gap with the current Blockchain networks. At the time when the PhD was started, only cryptocurrency-based applications existed. There is still a long way to go in this sense to be able to enter a wide range of new fields of application. However, new possible use cases and applications are already beginning to appear on the horizon, all of them, at the time of starting the doctoral thesis, are still only theoretical applications and very few practical applications or proofs of concept can be found.

Through the work of this thesis, we will try to dissect the key points why Blockchain networks are not yet established in the industrial world as well as to promote and study the feasibility of use cases in different industrial environments where applying Blockchain networks is a great advantage over the current state of the art.

2

Outline/Thesis Structure

The present document constitutes a thesis by a compendium of publications, which means it is formed by published papers as result of the research performed during the doctoral period. A brief description of each one of these contributions is provided below:

- **Publication I**: This work proposes an efficient integration of Hardware Security Module (HSM) and Blockchain technologies focusing on, mainly, public-key cryptography algorithms and standards, that result crucial in order to achieve a successful combination of the mentioned technologies to improve the overall security in IIoT systems. To prove the suitability of the proposal and the interaction of an IoT node and a Blockchain network using HSM a proof of concept is developed. Results of time performance analysis of the prototype reveal how promising the combination of HSMs in Blockchain environments is.

- **Publication II**: This paper proposes the design of Trusted Platform Module (TPM) virtualization in a container. To ensure integrity, different mechanisms, such as attestation and sealing, have been developed for the binaries and libraries stored in the container volumes. Through a Representational State Transfer (REST) API, the container offers the functionalities of a TPM, such as key generation and signing. To prevent unauthorized access to the container, this article proposes an authentication mechanism based on tokens issued by the Cognito Amazon Web Service. As a proof of concept and applicability in industry, a use case for electric vehicle charging stations using a microservice-based architecture is proposed. Using the EOS.IO Blockchain to maintain a copy of the data, the virtualized TPM microservice provides the cryptographic operations necessary for Blockchain transactions. Through a two-factor authentication mechanism, users

can access the data. This scenario shows the potential of using Blockchain technologies in microservice-based architectures, where microservices such as the virtualized TPM fill a security gap in these architectures.

- **Publication III**: This paper proposes a design of a secure sensor incorporating a Hardware Security Module with a Trusted Execution Environment in a hardware device based on a dual-core architecture. Through this combination of technologies, one of the cores collects the data extracted by the sensors, which implements the security mechanisms to ensure the integrity of this data. This proposed approach fits within the Blockchain networks acting as an Oracle. Finally, to illustrate the application of this concept, this paper describes a use case applied to wine logistics where this secure sensor is integrated into a Blockchain gathering data from the storage and transport of barrels.

Other publications have being included in this document, which were presented in conferences. These articles are:

- **Conference I**: This work presents a study on the integration of Blockchain technologies and HSMs, shedding light on the elements that these two technologies should have in common in order to be compatible, which are cryptographic curves and cryptographic standards, and highlighting the benefits they bring to the architecture where they are applied. This work has been published in the International Conference Computational and Mathematical Methods in Science and Engineering (CMMSE) on July 2022.
- **Conference II**: This work proposes a hardware implementation of a green certificate trading system based on Blockchain, in which prosumers use hardware secure elements to implement the cryptographic tools used to interact with the Blockchain. Smart contracts help to automate these processes, deleting intermediaries, saving costs and avoiding bureaucracy. This paper has been published in 18th International Conference on PhD Research in Microelectronics and Electronics (PRIME) on June 2023.

3

Funding and Methodology

In this chapter, the main methodological aspects related to the development of the different results achieved in this thesis are presented.

The Ph.D. research has been done while being part of the Research and Development Funding department at Infineon Technologies AG. The tasks of this contract consist not only to perform the technical work and demonstrators presented in this thesis, but also collaborating in the Proposal phase, Reports/Deliverables writing, and internal project management. The Ph.D. contract was attached to different EU and German-funded projects related to the research topic, as will be detailed in section 3.1.

3.1 Funding projects related to the Thesis

3.1.1 C4IIoT

"Cyber security 4.0: protecting the Industrial Internet of Things" (C4IIoT) is an European Funding project funded by the European Union's HORIZON 2020, receiving a budget of 5 millions of euros. C4IIoT builds and demonstrates a novel and unified IIoT cybersecurity framework for malicious and anomalous behavior anticipation, detection, mitigation, and end-user informing. C4IIoT framework provides a holistic and disruptive security-enabling solution for minimizing attack surfaces in IIoT systems, by exploiting emerging security software and hardware protection mechanisms, state of the art machine and deep learning and privacy-aware analytics, novel encrypted network flow analysis, secure-by-design IIoT device fabrication, and Blockchain technologies, to provide a viable scheme for enabling security and accountability, preserving privacy, enabling reliability and assuring trustworthiness within IIoT applications.

C4IIoT started in June 2019, with a duration of 36 months. The consortium is formed by 14 partners among which are IBM Israel, Centro Ricerche Fiat and HP Italy.

3.1.2 COLLABS

"COmprehensive cyber-intelligence framework for resilient coLLABorative manufacturing Systems" (COLLABS) develops, validates, demonstrates, and supports a comprehensive cyber-intelligence framework for collaborative manufacturing, which enables secure data exchange across the digital supply chain while providing high degree of resilience, reliability, accountability and trustworthiness, and addressing threat prevention, detection, mitigation, and real-time response.

COLLABS started in March 2020 with a duration of 36 months. The total investment for the project is 6 millions of euros funded by the European Union's HORIZON 2020. The consortium is compound by 13 partners among which are SIEMENS, Renault and Phillips.

3.1.3 tbiEnergy

"Trusted Blockchains für das offene, intelligente Energienetz der Zukunft" (tbiEnergy) project builds on the existing regulated energy market and addresses the acute gap of the lack of convenience services of today's smart grid through a holistic Blockchain approach. With Blockchain technology and smart contracts formulated in a Blockchain, innovative business models can be realised without high investments in information and communication technology (ICT) or software infrastructure while maintaining inherent security.

tbiEnergy started in June 2020 with a duration of 36 months. The total investment for the project is 1 million of euros funded by Bundesministerium für Wirtschaft und Klimaschutz. The consortium is compound by 5 partners including Infineon, which are Devolo AG, Hochschule Bremen, Arxum GmbH and Stadtwerke Trier AöR.

3.1.4 PROGRESSUS

"Highly efficient and trustworthy electronics, components and systems for the next generation energy supply infrastructure" (PROGRESSUS) project aims to introduce a next-generation smart grid demonstrated by the application example of a smart charging infrastructure integrating seamlessly into current smart-grid architecture concepts. To do so, it will research new efficient high-power converters that support bidirectional power flow. New DC microgrid management strategies for energy efficiency and service provision that consider renewable energy sources, storage and flexible loads will be

investigated. It will also explore novel sensor types, inexpensive high-bandwidth communication technologies and security measures based on hardware security modules and Blockchain technology to protect communication and services. The project's solution will promote a more environmentally friendly and efficient next-generation energy supply infrastructure.

PROGRESSUS started in April 2020 with a duration of 36 months. The total investment for the project is 19.5 million of euros funded by Electronic Components and Systems for European Leadership Joint Undertaking. The consortium is compound by 26 partners including Infineon, which are Devolo AG, Delf University, Pisa University among others.

3.1.5 EDGELESS

"Cognitive EDGE-cloud with serverLESS computing" (EDGELESS) project aims to leverage the serverless concept in all the layers in the edge-cloud continuum to fully benefit from diverse and decentralised computational resources available on-demand close to where data are produced or consumed. In particular, we aim at realising an efficient and transparent horizontal pooling of the resources on edge nodes with constrained capabilities or specialised hardware, smoothly integrated with cloud resources, which is a giant leap forward compared to state-of-the-art vertical offloading solutions where the edge is a mere supplement of the cloud.

EDGELESS started in January 2023 with 36 months of duration. The total investment of the project is almost 4 million of euros funded by Horizon Europe under the European Health and Digital Executive Agency (HADEA). The consortium is compound by 12 partners including Infineon, Telefonica, Siemens, University of Cambridge among others.

3.1.6 HardSec4IoT

This project proposes the generation of a public key infrastructure that allows the secure exchange of keys between IoT devices, as well as the development of a low-cost hardware platform capable of supporting this infrastructure. This platform will be based on a cryptoprocessor previously developed by the research team, which will provide each node with the following characteristics: Each node will have the capacity to generate secure random private keys, communications between nodes will be encrypted using AES, key distribution will be carried out using a public key cryptosystem based on elliptic curve cryptography, and a single group key will be used to optimise memory requirements and reduce message exchange. In accordance with the above, the following general objectives are proposed:

- O1. Development of a public key infrastructure for secure group key exchange in a heterogeneous network of IoT devices based on elliptic curves.

- O2. Development of an IoT node based on reconfigurable technologies as a low-cost platform to perform the proof of concept of the proposed public key infrastructure.
- O3. Integration of different types of IoT nodes, to test the performance and scalability of the developed proposal.

HardSec4IoT started in July 2021 with 24 months of duration. The total investment of the project is 25000 of euros funded by Programa Operativo FEDER 2020.

3.2 Methodology

The research line proposed for this thesis has been carried out satisfactorily. In this plan, a series of steps were presented to be carried out during the duration of this thesis. Both the study of the different IoT platforms and the Blockchain technologies have been exposed in the state-of-the-art publications that make up this doctoral thesis. As well as the integration of these two technologies (IoT platforms with HSMs and Blockchain) is explained mainly in the first publication. A concept of an IoT node that enables hardware security, both operating as a Blockchain client and operating as a Blockchain Oracle, has been proposed in several publications. Through the second and third publication the application of Blockchain technologies with hardware security in real environments has been demonstrated.

Thanks to the technologies provided by Infineon Technologies AG as well as open source software/hardware, the development of this thesis has been carried out successfully.

During the development of this thesis different components provided by Infineon have been used. Infineon offers a wide range of secure elements with a large availability of functionalities, as well as a wide range of microcontrollers. The main secure element used in this thesis is the Infineon OPTIGA^{TM} TPM2.0 which is designed according to the Trusted Computing Group (TCG) specifications and are certified Common Criteria CC EAL4+. The security functions include system and data integrity, authentication, secured communication, secured data storage and secured updates. In this thesis has been used as well Infineon OPTIGA^{TM} Trust M which is a high-end security controller that provides an anchor of trust for connecting IoT devices to the cloud, giving every IoT device its own unique identity. It offers a wide range of security features, making it ideal for industrial and building automation applications, smart homes and connected consumer devices.

Among the range of microcontrollers offered by Infineon, the PSoC64 has been used in this thesis. This microcontroller belongs to the PSoC6 family, which is based on an ultra-low-power 40 nm process technology and features a dual-core ARM Cortex-M4 and Cortex-M0+ architecture that allows programmers to optimise power consumption

and performance simultaneously. PSoC64 also provides a PSA Level 2 certification, which allows to developers of IoT systems have an extended level of trust, PSoC64 is equipped with the ARM PSA holistic set of threat models, security analyses, hardware and firmware architecture specifications. The PSoC 64 MCUs are ideal for cloud-connected applications that require protection of user data and trustworthy firmware updates.

Part II

Publications

4

Integration of Hardware Security Modules and Permissioned Blockchain in Industrial IoT Networks

---

ANTONIO J. CABRERA-GUTIERREZ[1,2], ENCARNACION CASTILLO[2], ANTONIO ESCOBAR-MOLERO[1], JOSE A. ALVAREZ-BERMEJO[3], DIEGO P. MORALES[2], and LUIS PARRILLA[2].

1. Infineon Technologies AG, Am Campeon 1-15, 85579, Neubiberg, Bavaria, Germany
2. University of Granada, Department of Electronics and Computer Technology, Faculty of Sciences, Granada, 18071, Andalucia, Spain
3. University of Almeria, Department of Informatics, Carr. Sacramento s/n, Almeria, 04120, Andalucia, Spain

Journal: IEEE Access

- Received 7 October 2022, accepted 25 October 2022, date of publication 28 October 2022
- DOI: 10.1109/ACCESS.2022.3217815
- Impact factor: Q2
- JCR Rank: 3.9, 100/275 Q2 in the category Engineering, Electrical & Electronic.

Abstract

Hardware Security Modules (HSM) serve as a hardware based root of trust that offers physical protection while adding a new security layer in the system architecture. When combined with decentralized access technologies as Blockchain, HSM offers robustness and complete reliability enabling secured end-to-end mechanisms for authenticity, authorization and integrity. This work proposes an efficient integration of HSM and Blockchain technologies focusing on, mainly, public-key cryptography algorithms and standards, that result crucial in order to achieve a successful combination of the mentioned technologies to improve the overall security in Industrial IoT systems. To prove the suitability of the proposal and the interaction of an IoT node and a Blockchain network using HSM a proof of concept is developed. Results of time performance analysis of the prototype reveal how promising the combination of HSMs in Blockchain environments is.

4.1 Introduction

Industrial Internet of Things (IIoT) collects and analyses data to deliver insights that help industrial organizations become more agile and making better-informed business decisions more quickly than ever before [1]. This leads to better quality control, and more efficient, streamlined supply chain management. It also benefits predictive maintenance, field service, energy and facilities management, and asset tracking.

In the Digitization of Everything era, security breaches are no longer even newsworthy. The spread of cloud services and the advent of the Internet of Things (IoT) have urged enterprises to enhance security and rethink their company policies. The overall complexity of a smart factory IoT system is extensive, and the number of security loopholes subsequently increases to a dramatic extent [2]. Clearly, traditional firewalls and antivirus systems will not be sufficient to protect complex IIoT infrastructures. An IIoT network requires an advanced security system, not only to ensure a non-disruptive smart factory workflow or to protect employees and assets, but also to secure business-critical information from competitors.

The information produced by the IIoT devices needs to be gathered and stored securely in specialised systems or hardware. Usually, for managing and processing this information, a client-server model is set up where dedicate machines, provide functionality to other programs or devices. These functionalities may include sharing data or resources between multiple clients, performing computation for a client using specialised software, or simply to gather and store information securely produced by the computational clients (in what respects to Industry 5.0, such actors might be the IIoT devices and robots).

Protocols used in IIoT typically are implemented using client-server (Zigbee [3] or LoRa [4]) or publish-subscribe (MQTT [5]) paradigms. In order to ensure enrollment and

communications in such networks, these protocols often include additional mechanisms to introduce security such as symmetric encryption, mainly based on Advanced Encryption System (AES) [6], or they can also include public-key cryptography through Transport Layer Security (TLS) [7]. Nevertheless, these types of architectures are prone to cyber-attacks [8].

The cost of cybercrime includes damage and destruction of data, stolen money, productivity losses, theft of intellectual property, theft of personal and financial data, embezzlement, fraud, post-attack disruption of the normal course of business, forensic investigation, restoration of harnessed data and systems, and reputational harm. In this context, industrial organisations pursuing to implant IIoT, the concern for a cyber-attack is not only focused on loss of data, but also on safety, integrity and availability of data and services. Consequently, the top four IoT security issues that need the greatest attention are authentication/authorization, access control, data encryption and the use of IoT devices as potential gateways to sensible systems [9].

Decentralized paradigms provide solutions to these needs by allowing data access control by different entities in order to enable auditability of events and policies, and to verify the integrity of all data items. Blockchain solutions are based on this concept [10], making use of cryptography to sign transactions or to add/remove nodes to/from the network. Distributed ledger technologies[11] (DLT) such as Blockchain, are based on maintaining distributed copies of a database which contains records of the transactions performed across the network. This scheme, along with a consensus algorithm previously agreed by all participants in the network for validating the transactions, allows reaching authenticity and immutability of those records [12]. However, these networks present serious scalability problems when the ledger is required to be updated and validated by a large amount of participants [13]. In order to avoid this issue, the number of participants in the network should be limited, or the traditional consensus mechanisms should be modified. The solution adopted by Blockchain networks designed as a support to currencies, as Bitcoin or Ethereum , where the transaction must be validated by 51% of the entities that makes up the network, does not provide a feasible solution to the scalability problem and requires high computing and energy resources. A more efficient proposal is the known as Permissioned Blockchain (PB) [14].

PB is a distributed ledger which is not publicly accessible. In this scheme, the participation of a member in the network requires certain permissions granted at registration time by Blockchain administrators through certificates. Hence, PB offers an additional security layer over typical Blockchain networks such as Bitcoin. Furthermore, PBs are compound by entities who require an identity and a role definition within the Blockchain.

Typically, the keys and certificates involved in a Blockchain are stored in a "software wallet" [15]. In the case of public-key cryptosystems, public-private key pairs have to be generated through random number generators (RNG) in order to follow cryptographic

standards. If these RNGs are implemented in software, the generated keys are also stored in software, thus becoming a security vulnerability [16]. Software-based security is not enough to protect systems as the stored data can be read, modified and distributed effortlessly. In order to avoid it, a hardware-based root of trust that renders embedded software trustworthy becomes necessary. In this sense, Hardware Security Modules (HSMs) offer a solution which relies on [17]:

- (I) High entropy random number generation.
- (II) Tamper-proof protection, by enabling secure storage of private cryptokeys and sensitive information. In this sense, HSMs are designed to guarantee inaccessibility of store information from external means, thus hindering physical attacks.
- (III) Keys backup and restoration.

In short, the existing problems in traditional architectures such as the centralization of resources can be mitigated by a decentralization of them using Blockchain technologies. Nevertheless, this introduces a new concern regarding how to protect sensitive data, since typically, these data are stored in software repositories. Protecting cryptographic material that is used intensively in Blockchain networks has become essential, thus making the use of HSMs sense.

The use of HSMs allows the storage and generation of the keys in a secure way. Thus, the combination of both components, HSMs and DLT technologies, offer a high robustness to the system in two levels:

- (I) HSM adds a new security layer –hardware-enabled security level– which impacts in the higher-level system security protocols.
- (II) DLT enables horizontal security –security between entities connected to the network in the same layer– in device-to-device communications. This level relies on the decentralized access control.

With the union of these two technologies, the problems of centralization and the protection of keys in software repositories are solved. This paper proposes the integration of HSM (focusing on Trusted Platform Modules (TPM)), and Blockchain, emphasising the key elements that make this integration possible, as well as the development of a proof of concept that demonstrates the suitability and performance of the communication between these two technologies in an IoT node.

The rest of the article is organized as follows: Section 4.2 presents an overview of the actual communication paradigms and the way they manage cryptography used in the context of IIoT networks. Section 4.3 describes HSMs, focusing on TPMs, and PB

technologies, focusing on Hyperledger Fabric. In Section 4.4, there will be a discussion about the different key components which are essential to integrate both technologies, mainly public-key cryptographic algorithms and standards (PKCS). Furthermore, a proof of concept is presented, showing the different interactions between IoT nodes and the Blockchain network. This section also presents a performance test which shows the capabilities of TPM when operating on an IoT node in a Blockchain network. Furthermore, a security analysis is carried out showing the attacks that this architecture prevents. Finally, Section V summarizes the conclusions, emphasizing the benefits of the union of these two technologies.

4.2 State of the art

Current security paradigms on computer networks are based on Public Key Infrastructures (PKI) [18]. In this type of paradigms, the security of the overall system relies on a Certification Authority (CA), thus presenting some issues inherent to centralization: on the one hand there is a Single Point Of Failure (SPOF), and in the other hand, every attack will be directed to CAs, which will require an extremely high level of security. Also, Denial of Service (DoS) attacks can be performed more easily [19], and the requirement of user identification for registering in CAs implies lack of anonymity and privacy.

Since the emergence of IoT technologies, especially in industrial networks, the development of new security paradigms has become a need. In this scenario Blockchain is considered the most relevant technology for introducing a decentralized security system in IIoT networks. The combination of IIoT and Blockchain offers a trusted system where the information is reliable and can be traceable. Data stored in a Blockchain ledger remains immutable over the time as well as the sources remain identified at any time.

In a typical IoT system, registration and authentication data are stored in a central entity. These data are required for the registration of new IoT nodes, but the need of this central entity introduces some vulnerabilities, as has been previously carried out. One solution to this issue consists on decentralising this architecture as shown in Fig. 4.1, where the database corresponding to the central entity is replicated in different client nodes. After decentralization, an attacker has more difficulties to perform unauthorized modifications in the database, because the different clients have to approve these changes. As will be described in next subsection, Blockchain enables the implementation of that decentralized infrastructure.

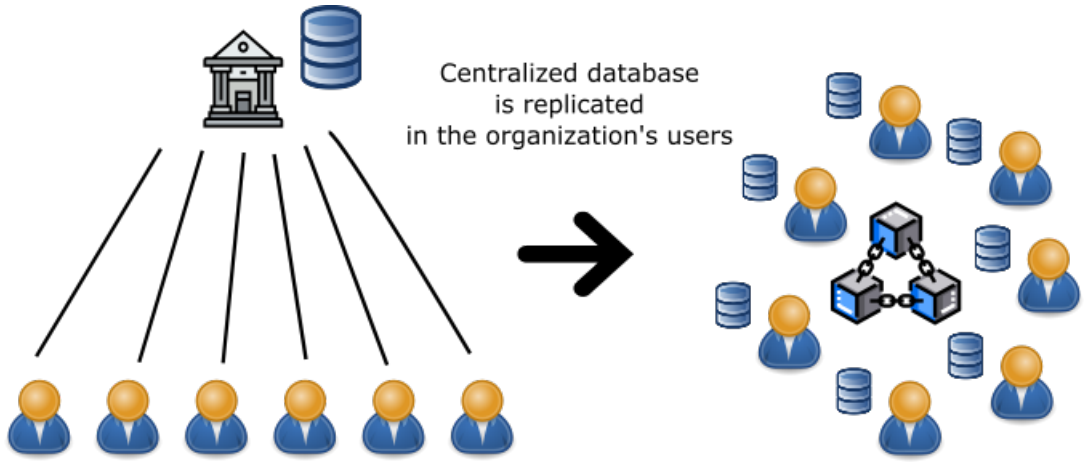

FIGURE 4.1: Typical PKI infrastructure vs. decentralized ledger infrastructure.

4.2.1 Blockchain networks

Blockchain technologies are being applied to multiple fields [20] [21] [22], although its application to the IIoT scenario is relatively recent. In this field, Blockchain has a lot of potential use cases as automotive and mobility [23], consumer applications [24], tracking logistics [25], supply chains [26], energy [27] and health [28]. In all these cases, Blockchain acquires greater relevance due to the participation of different organizations in order to complete a given process. These organizations must get to an agreement that will be later translated into policies implemented in the Blockchain in the form of smart contracts, which reflect in source code the existing relationships among the organizations involved. The execution of these contracts is ensured intrinsically.

In recent years, several Blockchain platforms have been developed which are subject to continuous changes. One of the most popular is Ethereum [29], which is the first open-source Blockchain platform introducing the concept of smart contracts. As it is well known, smart contracts enable a lot of new applications of Blockchain beyond cryptocurrencies. Smart contracts are an useful feature for IIoT, but in the case of Ethereum, some characteristics prevent from being used in the IIoT use cases. Basically, its consensus algorithm is based on Proof of Work (PoW) [30]. This process requires a high amount of computing resources to avoid attackers to modify the Blockchain, while users generating a new block are rewarded with cryptocurrencies. This consensus algorithm is not applicable to an IIoT environment because the nodes that make up the network are usually low-power consumption devices, and they are not designed to perform such large computations, but for data collection and even run some control algorithm. There are other types of consensus mechanisms such as the Proof of Stake (PoS) [30] which has lower computing requirements, but in contrast, this consensus mechanism requires that the entity who wants to participate in the network must make use of cryptocurrencies, i.e, all the entities in the network must have at least a small amount of them.

The use of smart contracts in Blockchain networks can generate vulnerabilities at the application level that an attacker can exploit. While this can lead to a problem, the use of HSM at the hardware level helps to mitigate these vulnerabilities by restricting the attacker's attackable surface to higher layers of the application.

There are other Blockchain platforms such as Multichain [31] or Quorum [32], which are different variations of Bitcoin and Ethereum, respectively. Multichain is a private Blockchain as opposed to its counterparts, which are public. Quorum is the permissioned version of Ethereum, what means that the participants of the network have some restrictions and they play different roles in the interaction among them.

In order to avoid the issues associated with the use of cryptocurrencies, there are other Blockchains that do not use cryptocurrencies, thus making them more adaptable to IIoT use case. One example is Hyperledger Fabric [33] which will be discussed further.

4.2.2 Hardware Security Layer

Blockchain introduces useful features for building a decentralized infrastructure, but it also introduces a significant security risk regarding the storage of keys in the different nodes. Usually these keys are kept in a repository, but not only that, they are also generated by software. This can lead to a major security breach, since algorithms used to generate required keys could be vulnerable to different attacks. Concretely, Pseudo-Random Number Generators (PRNG) commonly used for generating keys are vulnerable to key replication if seeds are predictable or not properly randomized [34] [35].

In the particular case of IoT devices these issues arise intensively because it is not easy to have good sources for generating true random values required for the seed, being firmware-generated random values not enough for guaranteeing secure keys generation. Therefore, for having a reliable entropy source in these devices it is necessary to generate random values directly from physical sources. Then, a feasible solution is to use the well known True Random Number Generators (TRNG), which generate true random numbers from high entropy microscopic physical events in the hardware as statically noise of signals, photoelectric effect or quantum phenomena [36]. In this way, HSMs offer TRNGs which make it ideally to be used in devices that interact with the Blockchain. HSMs hinder side-channel attacks in the sense that when the key is generated inside the chip, the attacker cannot know the time the chip has taken to create the key internally, and there are specific countermeasures against the analysis of noise, electronic leaks and power consumption [37] [38].

Regarding the storing of the generated keys, they are usually stored in repositories, which it is a big risk, as commented before. Indeed, the keys can be easily extracted, manipulated and replicated by an attacker.

A little bit more robust method to protect the keys by software is a Trusted Execution Environment (TEE) [39]. TEE is a standard that creates an isolated environment which runs over or in parallel with the operating system. A TEE guarantees the authenticity of the executed code, the integrity of the runtime states and the confidentiality of its code, data and runtime states stored. Thus, TEE provides secure enclaves in order to execute and store sensitive assets and critical data such as private keys.

Although a TEE enabled system resists software attacks, it is still vulnerable to kernel faults, side-channel and physical attacks, which can be performed in order to undermine the isolated environment [40]. Furthermore, in the case of IoT devices TEE can not be implemented, because they usually run firmware without any operating system support.

This issue can be overcome using an HSM, because once the key has been generated, it can never be extracted, thus providing a secure storage for generated keys. In this way, HSMs present tamper resistance which avoids physical attacks such as probing attacks where an attacker sets a probe on a wire and reads the signals being transmitted over the wire during chip computations [41].

As discussed above, the Blockchain platforms and hardware security layer technologies have drawbacks and security issues that can be exploited in certain scenarios, such as extracting the cryptographic keys from a TEE using hardware attacks or failing to properly protect cryptographic material used in a Blockchain network. This is why this article brings together the combination of HSMs, focusing in TPMs, and PB networks offering together the benefits of each technology, thus providing an IIoT architecture that incorporates different layers of security, proposing a more robust system on the technologies analysed in this section.

The next section will discuss in more detail the benefits of HSMs, in particular TPMs as well as PB technologies focusing in Hyperledger Fabric.

4.3 Background

Before starting to discuss about the integration of the two components and the proof of concept presented in this article, the components that make up the proposal of this paper should be introduced, focusing especially on TPMs and Hyperledger Fabric since the proposed integration cannot be understood without an in-depth knowledge of the characteristics offered by these technologies.

4.3.1 Hardware Security Modules

HSMs are being extensively used for device protection, providing a secure framework for authentication and identification. An HSM consists of a cryptographic processor

which implements in hardware different cryptographic algorithms required for these tasks. In this sense, it offers tamper protection against harmful manipulation and strong authentication mechanisms [42].

HSMs usually are delivered in different form factors. Typical ones are security cards, widely used in people identification, and chips installed in a PCB which are connected to the CPU of the system under protection. The HSM will be required to perform different security tasks: signing, signature validation, encryption, decryption or hashing, as well as secure storage and trusted random number generation. In short, an HSM provides a root platform of trust [43].

TPMs are a subgroup of HSM devices, whose features are defined by the Trusted Computing Group (TCG) [44]. In the next section it will be explained in detail.

4.3.1.1 Trusted Platform Modules

TPMs are standardized by the TCG. Being TPM2.0 [45] [46] the latest specification. In all of them, a TPM is defined as a hardware device including volatile and non-volatile storage, and a set of cryptographic algorithms implemented in hardware. In addition, the different TPM standards include an API specification to interact with the TPM [47]. In short, a TPM is a hardware component that provides secure storage of cryptographically protected data (keys, certificates, passwords and other data related to the internals of the TPM as Platform Configuration Registers) and enables the generation of keys inside the TPM using TRNG, private/public key encryption and signature operations. Fig. 4.2 shows the architecture of a TPM2.0 device [45], where the different components are interconnected by a bus, which also connects to the I/O interface. In addition to the aforementioned TRNG and the non-volatile memory, where the Platform Configuration Registers (PCR) [48] are located, there are other important modules such as the symmetric and asymmetric key engines and the key generation engine.

Typically, TPMs are used in computing systems supporting a BIOS or a similar firmware in charge to boot the device for adding a security layer below the software. Indeed, keys generated in the device cannot be extracted outside of the TPM, hence, data secured by these keys will be not exposed. In this sense, the TPM provides a root of trust. In this scheme, PCRs are records containing a concatenation of hashes [48] which are the base of the different protection mechanisms performed by the TPM. As an example, during the secure boot process provided by modern BIOSes, the startup firmware checks different parameters of the system as the peripherals attached, the status of the memory, and others. These parameters are hashed and compared to the ones stored in PCRs. If the result of these comparisons is correct, the system will boot successfully. The hashed values are usually critical parameters of the system, thus preventing the system from booting if any modification is detected [49]. Another important feature provided by

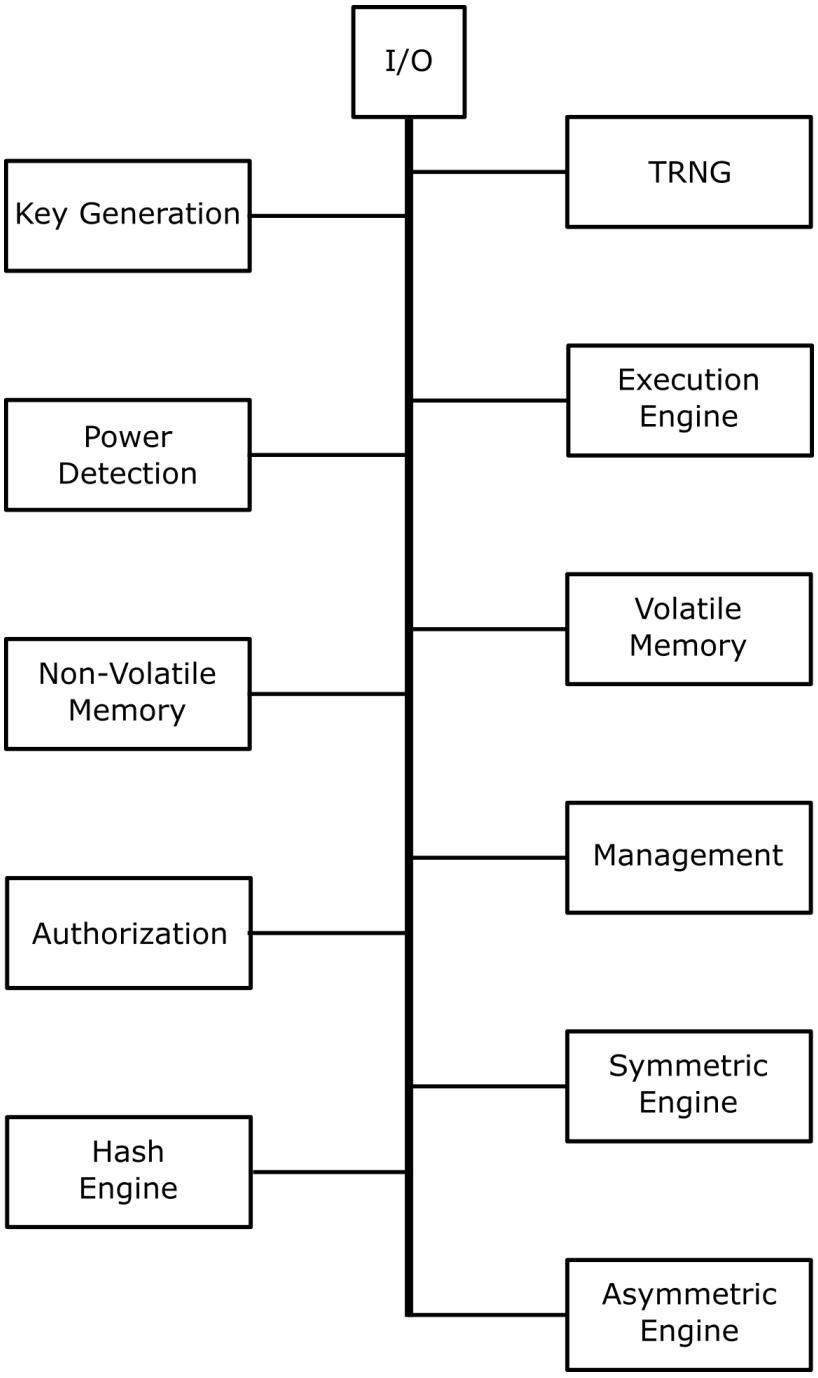

FIGURE 4.2: TPM2.0 architecture.

PCRs is attestation, which is the ability of proving authenticity in the system using a public-key cryptosystem with an Endorsement Key (EK) [50]. In a typical attestation scenario, the PCRs values along some piece of code or executable are signed with the EK in the TPM. In order to verify the authenticity of the code, the signature has to be verified jointly to the PCR values [51] thus guaranteeing that the code which is running in the system is totally secure.

PCRs are also involved in another operation known as key sealing. When the measured

system parameters correspond with the values stored in the PCRs, it is possible to unseal a key used for encrypting data in the system. Without this key, data cannot be decrypted, preventing the access to protected data if the system is not in a safe state [52]. Furthermore, the process of encrypt/decrypt used with the TPM is called binding/unbinding since the data encrypted using a key in the TPM can be only decrypted using the same key of the TPM. If this key is used in other TPM the data cannot be decrypted/unbounded.

In conclusion, the TPM can perform different cryptographic operations and results in a root of trust from which different operations can be carried out with security guaranteed by the TPM. In fact these operations can also be performed by a normal HSM, but the main advantage of a TPM is that these operations are supported by the TCG, which ensures that all these operations are in the current state of the art and are standardized.

4.3.2 Permissioned Blockchain technologies

As it had been commented before, Blockchain technologies offer a new paradigm that comes to replace the typical PKI schemes. However decentralized solutions built on Blockchain technologies have difficulties to scale to the amount of devices and data aggregated by IIoT. This is one of the reasons why PB technologies appear [53].

These technologies offer more versatile consensus protocols and allow the enrollment of a limited amount of participants in comparison with public and no-permissioned Blockchains. This relies on the fact that it eliminates the unnecessary computation required to reach the consensus protocol. In this scheme, participants in the Blockchain network have different permissions to read, access and write information, while the configuration of the Blockchain is defined by the policies of the network, typically agreed by the members participating in it. As a consequence, the policies defined within a PB affect the behavior of all the members. In any case, this dependency on the policies does not affect the classification of the defined Blockchain in terms of being public, private, public permissioned or private permissioned. What really makes the difference is the maintenance of the identity of each participant in the Blockchain. This means that an entity can participate in the Blockchain network if, and only if, it has been previously certified as acting as itself by a CA. This is why the approach of the PB networks are considered as hybrid, since there is a central authority (the CA), which is the one that gives the authorization to be able to participate in the network [54], thus not being totally decentralized as in the permissionless networks.

This is the main feature that makes permission-based Blockchain networks so popular in the industry since security, identity and a defined role for each network participant are required. There are several cases of use in the industrial field [55] [56] that utilize this type of technology. In order to provide some examples, the most typical one is the

traceability chain, which must ensure that the characteristics of some consumer product are kept intact through the logistic chain. In this case, the agencies involved in this business model would be the intermediaries/logistics service, producers and consumers. Another example would be the supply chain, which would involve as many logistics services as banks and sellers/buyers. These are just some scenarios, but there are more cases emerging, as the energy exchange which is another scenario that is taking more importance with the time [57]. To carry out the implementation of these use cases there are whole suites and frameworks dedicated to the implementation of PB technologies, being Hyperledger Fabric one of them.

4.3.2.1 Hyperledger Fabric

Hyperledger Fabric (HLF) [33] [58] is a PB with support for executing smart contracts. HLF allows organizations to collaborate in a Blockchain establishing different roles and entities. Each node has its own function depending of its role that carries it out. HLF defines four different nodes, which are:

- HLF peer nodes, used to store a copy of the ledger, to endorse new transactions by invoking smart contracts, to commit new blocks into the ledger and to allow querying the ledger.
- HLF ordering nodes, used to create and distribute new blocks of transactions to the HLF peers. The organization that owns this node will be the one that creates the network and establishes the policies that govern the network.
- HLF clients, used to communicate with peer and ordering nodes in order to query the ledger and to propose new transactions.
- HLF Certificate Authorities, which issue certificates to administrators and network nodes.

In addition to this, a consortium must be defined specifying which organizations will participate in the network. These organizations communicate through a channel. This offers great versatility since an organization can participate in several channels with different organizations at the same time. For example, in the case of the supply chain, the logistics service can participate in one channel with the buyer and in another with the seller, but the buyer and seller do not have a common channel.

4.3.2.2 Hyperledger Fabric operations

In order to start the network, a client needs to execute some function contained in the smart contracts. When a function is triggered, an operation is done in the ledger: *write*

or *read*. When an operation is executed the endorsement policy comes into play. It describes which organizations must approve transactions before they will be accepted by other organizations onto their copy of the ledger.

When a transaction proposal takes place, that is, the client initiates it, the process to be carried out is the following:

- The peers verify that the transaction is well formed and that it has not been done before, avoiding replay attacks. It is also verified that the future transaction satisfies the policies of the channel.
- The transaction proposal executes some function of the smart contract.
- A response is produced which will be reflected on the state of the ledger if it is a write operation (or not if it is a read operation).
- The application disseminates both the transaction proposal and the response within a message to the peers for them to verify them.
- Once the transaction is verified following the channel policies, each peer updates its ledger and the status of the database.

Another operation that typically takes place in HLF is the enrollment process of users to the network. This operation is performed in the client, which has to contact the CA when it wants to enroll. Then, the CA issues a certificate as a result of the enrollment operation.

HLF defines two types of enrollment depending on whether the client to register as an administrator or a normal user. In the case of to be registered as administrator it is required to provide first some credentials (username and password) which must be already configured in advance in the CA. Then, the client makes a Certification Signing Request (CSR) where he attaches those credentials and, as a consequence of that, the CA returns a valid certificate if registration has result successful.

In the case of a registration as a user, the process is similar except that it requires the previous enrollment of an administrator which serves the user to register in the CA. Once this is done, the process of enrollment is the same as for an administrator. So in conclusion, in the administrator there is a process of enrollment and in the user a process of registration and enrollment.

As it can be seen, within the operations to be carried out in the Blockchain there are different cryptographic operations which should be performed in the TPM. In the next section a detailed integration between these two components is provided, Hyperledger Fabric and TPM, describing the operations carried out by a Blockchain network.

4.4 Integration between Hyperledger Fabric and Trusted platform module

Every operation related to the DLT requires cryptography [59], the need of secure hardware support by means of an HSM is a promising solution to the potential vulnerabilities exposed by relying just on software.

4.4.1 Elliptic Curve Cryptography

Key generation algorithms are involved in the generation of cryptographic keys. The size of these keys is directly related to the memory resources required for storing them, and the corresponding certificates and digital signatures. In the past, RSA [60] was the preferred Public Key Cryptosystem (PKC), but the updated computing capabilities of attackers requires a continuously increasing size of the RSA keys, which represents a problem for processors with limited computational and energetical resources [61]. In this context, Elliptic Curves Cryptography (ECC) has emerged as an alternative to RSA for PKC, as it provides a similar level of security to RSA, but requiring smaller key sizes.

One of the main issues with ECC is the selection of a secure curve for cryptographic applications [62]. The choice of the curve determines the parameters that lead to its efficiency and security strength. The main curves that are used in ECC algorithms are Weierstraß, Montgomery and Edward Curves. The National Institute of Standards and Technology (NIST) recommends the Weierstraß curves, whose general equation is:

$$y^2 = x^3 + ax + b \tag{4.1}$$

NIST establishes different recommended curves over binary and prime fields. Some examples of NIST curves defined over prime finite fields are:

- P-192, also known as secp192r1 and prime192v1.
- P-256, also known as secp256r1 and prime256v1.
- P-224, also known as secp224r1
- P-384, also known as secp384r1.
- P-521, also known as secp521r1.

In the case of Blockchain, the first curve used was the secp256k1 [63], also known as the “Bitcoin curve”. This curve is used also in Ethereum. The prime256v1 curve has been recommended by the NSA for use in government affairs. However, due to the

boom in quantum computing, this recommendation has been updated by proposing that the curves to be used for greater safety should be the secp384r1 when quantum computing reaches a more advanced stage. For example, the secp384r1 curve offers 192 bits of security instead the more commonly used curves like prime256v1 which provides 128 bits [64]. In the case of Hyperledger Fabric, the supported curves are prime256v1, also known as NIST-P256 curve, secp384r1 and secp521r1. For the HLF network to be compatible with an HSM, it must support the same or at least one of these elliptic curves. In this paper we have tested the compatibility with a TPM, concretely, with the Infineon OPTIGA^{TM} TPM2.0. Both Infineon TPM and HLF shares the NIST-P 256 curve.

4.4.2 Public-key cryptographic standards

Another key point to make possible the integration of these two technologies is the set of cryptographic standards implemented by them.Indeed, standards in cryptography establish the mechanisms and protocols to implement cryptographic algorithms in a secure way while facilitating encrypted data interoperability. Also, if a security breach is discovered, the corresponding standard is discarded and replaced by another. This results in the fact that the standards used are always being updated.

Cryptographic algorithms and protocols are standardized by different organizations dedicated to this purpose, some of them being public, and other private. Examples of public entities are NIST [65] IEEE [66], while RSA security LLC is a private company that has issued a set of standards called Public Key Cryptographic Standards (PKCS). Some of these PKCS standards have been abandoned or withdrawn, but others are in use today. In the case of PKCSs being used both by TPM and HLF, these are the PKCSs concerning communication between them. On the one hand we have PKCS10 [67], which it is also known as CSR. This PKCS specifies the format that messages sent to a CA must follow in order to authenticate the device on the network. This PKCS comes into play when a user or administrator needs to be registered on the HLF network. Once the CSR has been successfully requested, the CA issues a certificate in X.509 format. This certificate is stored in the device, either in the TPM or directly in the client's memory. A CSR contains the applicant's public key and data that acts as a proof of device identifier. Both the public key and the data are signed by the CA's private key, which generates an X.509 certificate, which is sent back to the applicant. On the other hand, the PKCS11 [68] is the standard which defines a standard method to access cryptographic services from tokens/devices such as HSMs, smart cards, and others.

In this sense, PKCS11 isolates an application from the details of the cryptographic device. That means, the application does not have to change to interface to a different type of device or to run in a different environment; thus, it enables the application to be portable. In our HLF-TPM integration proposal, PKCS11 standard will work as

an interface between the TPM and HLF. Indeed, supports PKCS11 standard in order to facilitate integration with HSMs. By default, HLF uses a software wallet in order to store the cryptographic material. Adding the correspondent configuration in HLF, we can change this storage method by a hardware storage. The PKCS11 standard is implemented in the TPM by the TCG group, therefore all the functions and methods that it contains inside are tested and proven using different applications, and we will use this standard as a hinge between these two technologies. The corresponding API is implemented at the highest level within the TPM software stack (TSS), so it abstracts the application on top of it, making it independent of the type of Blockchain network used, of all the functionalities implemented at a lower level in the TSS and of the TPM itself. Then, any TPM that includes the TSS in its implementation will be compatible with applications including the PKCS11 standard.

Having discussed the two main keys to the integration of these two technologies, we will now proceed to explain the different operations carried out between Hyperledger Fabric and TPM2.0. The idea of choosing TPM2.0 as the hardware support element lies in its ability to work with systems that implement operating systems as it enables operations such as trusted boot and remote attestation that other types of HSMs do not have.

4.4.3 Operations performed in Hyperledger Fabric using TPM2.0

In this section we will discuss how some critical HLF operations may be performed jointly with a TPM. These operations, which are traditionally carried out using software tools, which, as explained above, generate security breaches since they can be easily attacked, are executed internally in the TPM. These operations will be the enrollment of new users and administrators, and the signing of transactions. In the following, it is assumed that an HLF client is implemented on an edge node that collects data from an industrial environment. This data is periodically sent to the Blockchain network which is running distributed across different entities, CAs, peers, etc. Fig. 4.3 shows the proposal of an enrollment mechanism for the HLF client equipped with a TMP2.0 HSM interacting with a CA of the Blockchain network.

When a client wants to enroll in an HLF network, it has to initiate an enrollment handshaking with the CA. This process is the same whether it enrolls as an administrator or as an user, with the exception that the user has to be previously registered by an administrator. As shown in Fig. 4.3, in this operation the message sent to the CA contains the CSR and a user and password. CSR fields contain information concerning the identity of the client in accordance with the standard. In addition to the CSR, two fields are also included, the user name and password, parameters that have been previously stored in the CA. To successfully complete the enrollment process, the username and password sent have to match with the stored values. In the case of user enrollment, the administrator must have previously registered the user. Therefore, the registration

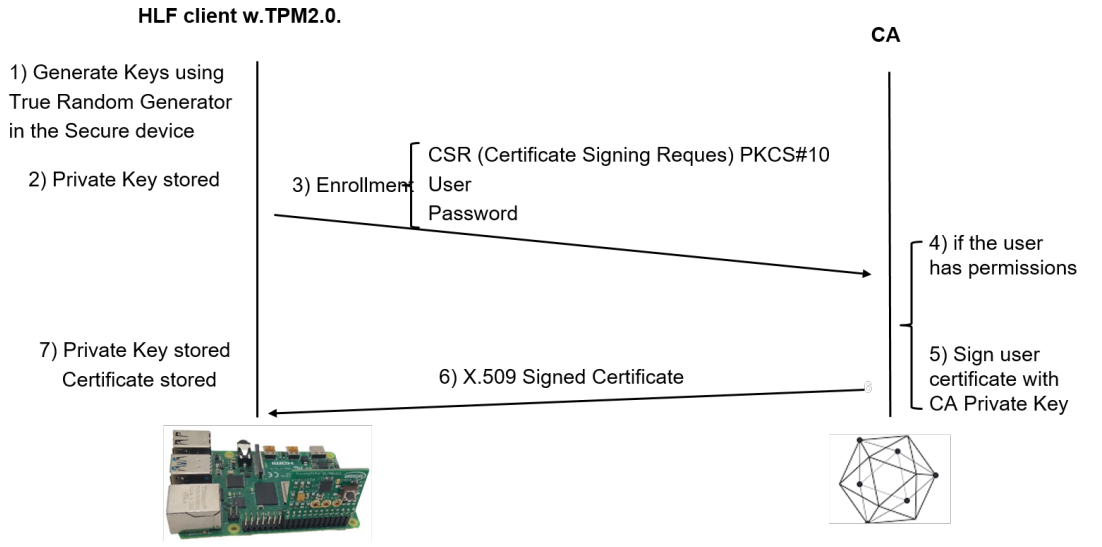

FIGURE 4.3: Enrollment mechanism.

process basically involves sending the user's username and password to the CA. The CA returns a secret and it is used by the user in the enrollment process instead of the username and password.

Note that the main difference between the two processes is that the registration process of the administrator has been previously completed. This may be because the system is deployed with predefined administrators or because it has been initialized through other mechanisms, such as sending the administrator's parameters through another medium, such as through credentials stored on a physical device or through other protocols. Once the CA has received the CSR with the corresponding parameters, the CA signs the CSR with its private key and generates a X.509 certificate as a consequence. This certificate will be sent back to the client, who can store it in the TPM or simply in the device's memory. This certificate will be in charge of validating the communications later, when the client interacts with the rest of the Blockchain network.

Fig. 4.4 shows the signature procedure when an HLF client interacts with other peer node, where the TPM performs the required cryptographic operations. Note that this direct interaction between peer nodes is only possible if the clients have been previously enrolled into the Blockchain network.

In this situation, transactions are triggered by the HLF clients. These clients, typically, include sensors producing data that is uploaded to the ledger, thus generating a distributed copy of this data. In order to upload and generate this distributed copy, the clients have to start a transaction. In the case of HLF, the fields in a transaction are:

- Header: Including the metadata of the transaction.
- Signature: A cryptographic signature of the transaction hash.

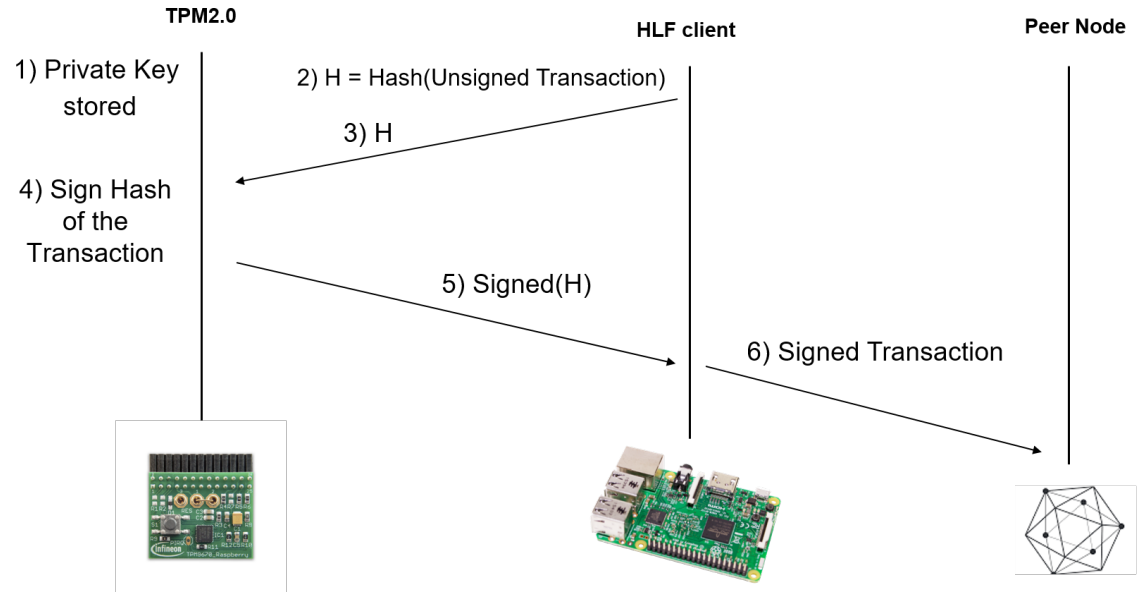

FIGURE 4.4: Signature procedure mechanism between the TPM2.0 and the Peer node.

- Proposal: The input of the smart contract which results in the data to update in the ledger.
- Response: The output of the smart contract; if the transaction is validated successfully, the output will be added to the ledger.
- Endorsements: A list of different endorsements nodes which have to validate the transaction fulfilling the endorsement policies of the network.

These fields form a data structure that is hashed and then signed by the client. Then, this signature is included in the corresponding field, and sent to the HLF network, thus forming a chain. It should be noted that this operation, which is typically executed in software, in our scheme is carried out internally in the hardware included in the TPM. Next, the request is received by the peer nodes which are in charge of validating the transaction and applying the necessary changes on the ledger, in the case of being required. Regarding the transaction verification process, it is carried out by other nodes as the Orderer and peers. In this process, the different nodes verify the transaction signature by means of the client's public key. Note that this process is not internal to the TPM, and it is not carried out within the HLF client where the TPM is located.

Basically these are the main processes involving the TPM in an HLF client. Of course the TPM can act on the other nodes, both CA and peers. The next subsection presents a proof of concept of the proposed application of TPM to HLF networks.

4.4.4 Proof of concept

In order to demonstrate the viability of combining TPM2.0 and HLF in an IoT node, a proof of concept has been developed. The scheme of the demonstration is shown in Fig. 4.5, where the interaction of an IoT node with a simulated HLF Blockchain network is presented. The IoT node is a Raspberry Pi 4 Model B (RPI) with an Infineon OPTIGA^{TM} TPM2.0 attached. The RPI runs a HLF client which interacts with the other nodes being part of the Blockchain network. In this proof of concept, a default HLF network configuration is used, as the objective is to show the feasibility of inter-operation between HLF and TPM and not a deployment of entities for a specific use case. The HLF network includes two organizations and two peer nodes per organization. Each organization has its own CA. There is also an HLF Orderer node, which performs transactions ordering [58] and maintains the list of organizations in the network. Each of these entities runs in Docker containers [58] in a virtual network hosted on a separate computer. This environment shows a scheme of two organizations which are part of a consortium in which a client, that belongs to one of these organizations, wants to register in order to be able to send data to the Blockchain as transactions.

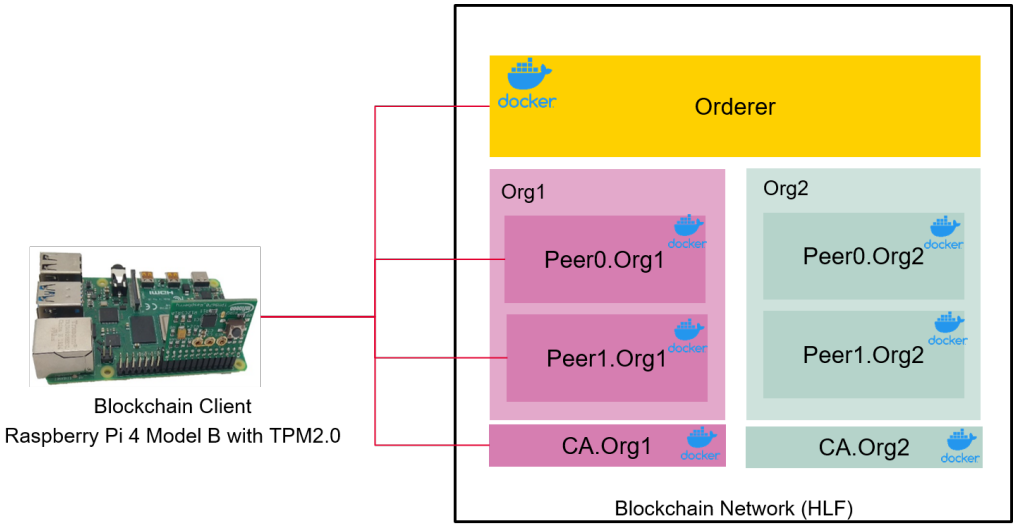

FIGURE 4.5: Proof of concept, Raspberry Pi 4 Model B with Hyperledger Fabric.

With these elements, this proof of concept allows to show the feasibility of the integration between HLF and TPM2.0, as well as that the mechanisms performed in a Blockchain network from the point of view of a client, can be perfectly integrated in an IIoT network. These mechanisms, both signature and enrollment process, are the ones in which the TPM actively participates in. The implementation of these mechanisms into the TPM creates a root of trust along the Blockchain network. Nevertheless, the possibility of integrating these mechanisms using different firmware or variants in the TPM can lead to different time performances. This aspect will be discussed in the next section.

This proof of concept includes at least two peer nodes per organization to make it more realistic. For validating a transaction made by the client at least two nodes, one from each organization, are required. The interaction between the HLF client and the Blockchain network starts with the enrollment process. As commented previously, the enrollment process in HLF is different depending on the role of the client, which can have the role of administrator or user. In order to enroll a user, the client must enroll an administrator before. The administrator plays the role of registering new users who will be in charge of querying and updating the ledger.

This mechanism is shown in Fig. 4.6, and starts with the enrollment of the administrator. Note that the CA must have previously initialized some credentials (name and password) in order to allow the operation. When the enrollment process is finished, the administrator receives the certificate issued by the CA. For user enrollment, the user must first register with the CA through the administrator, after that, it returns a token called "secret" that can only be used once for the user to enroll. After this process the user receives a certificate which will be stored.

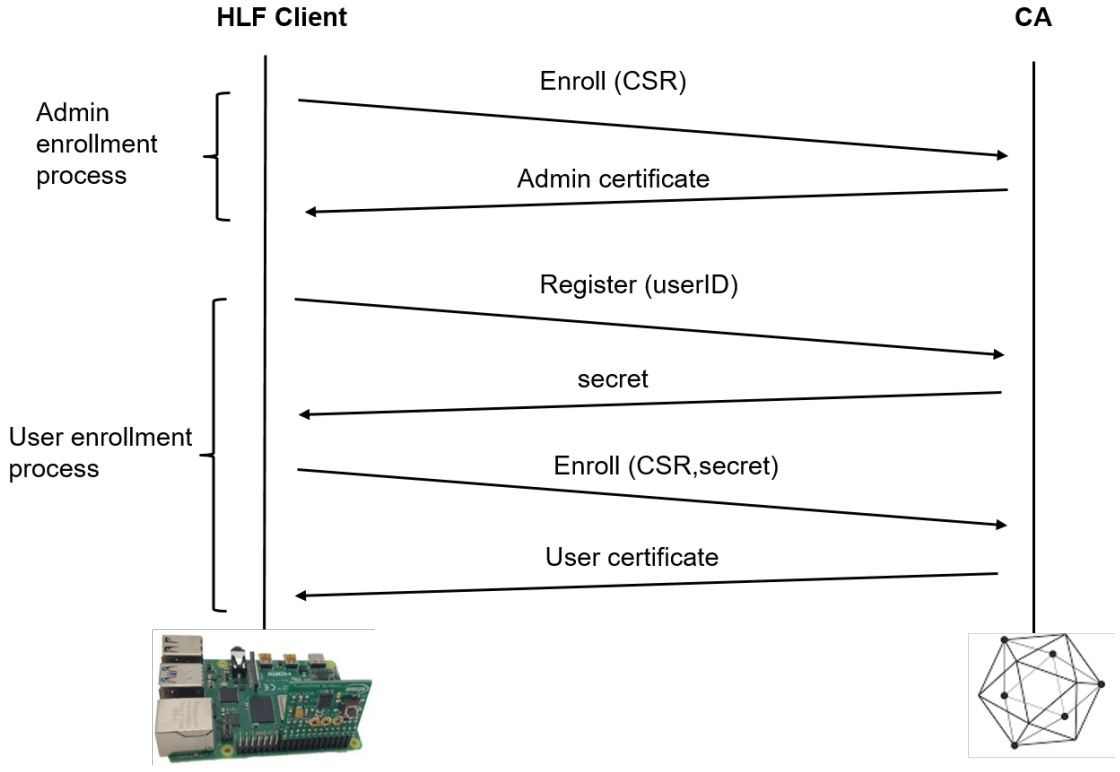

FIGURE 4.6: Admin and User enrollment process.

Once this process is completed, the TPM of the RPI will store both the user and administrator keys as well as the certificates issued by the CA. The user is in charge of interacting with the ledger by executing functions defined in the smart contracts running on the peer nodes. These operations can be either query or update operations. Operations involving a write, and therefore, an update to the ledger must be performed through transactions signed by the client and verified by the peer nodes.

In Fig. 4.7 it is shown how the client initiates a transaction sending a transaction proposal to the peers. The peers perform the endorsement service which is in charge to execute the smart contract and obtain a proposal response with the output of the smart contract and the endorser's signatures. Then, the client receives the proposal response and validates it. The client builds a transaction and sends it to the Orderer. This transaction includes the transaction proposal, transaction response and the endorsements. The Orderer validates the transaction and creates a block which contains the validated transactions and broadcast this block to all the peer nodes. The peers execute the transaction and update the state of the database decentralized in the peer nodes. The block is finally committed in all peer nodes.

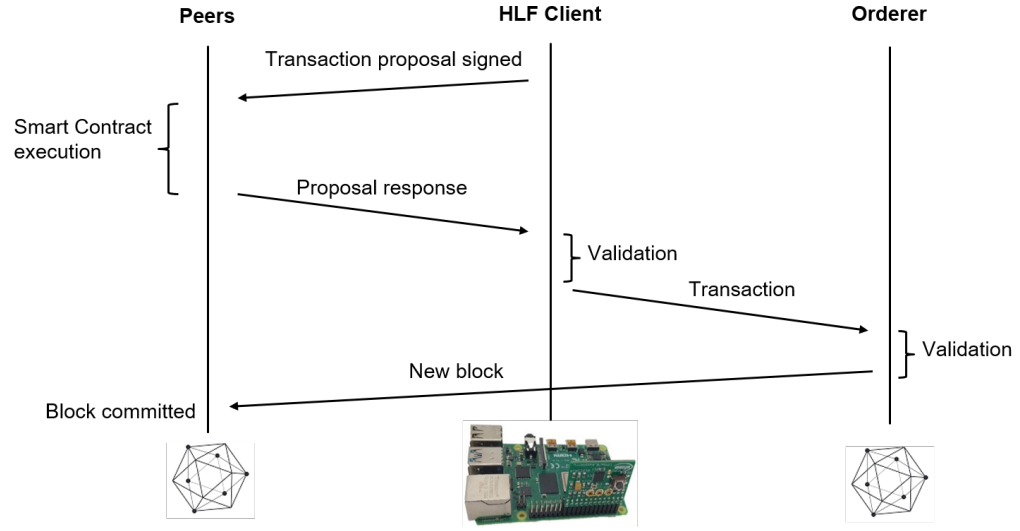

FIGURE 4.7: Transaction mechanism between Client node and the Peer nodes.

Both the enrollment process and the transaction process are the two main mechanisms that occur in HLF. These processes are initiated by the client, which makes use of the TPM to generate the keys and sign them with the private key. This proof of concept is intended to illustrate what the complete process would look like using an IoT node. On this basis, it is possible to build much more complex use cases that encompass real use cases. The next section shows the results obtained from this proof of concept in order to discuss its application in IIoT environments.

4.4.5 Performance analysis

The use of TPM in IoT nodes has certain limitations compared to an HSM, Trusted Execution Environments (TEE) or cryptographic software. Normally an HSM is dedicated to be a hardware accelerator for cryptographic operations. This implies that the time to perform an operation in the TPM is longer than if it were done in an HSM. Otherwise, the gain in security that a TPM offers makes it advantageous depending on the scenario.

In this work a benchmark has been performed comparing different approaches. To evaluate this time comparison, an average has been made over 100 samples. The operations to be measured are:

- Generation of the keys: Generation of a NIST-P265 private-public key pair using the TPM.
- Signature: Execution of the signature algorithm using prime256v1 elliptic curve and SHA-256 hash function.
- Verification of the signature: It is carried out using the public key and the signature previously computed.
- Commissioning of the client: The commissioning encompasses both key generation and the CSR, i.e. the signing of the client's public key by the CA in order to issue the certificate to the user. The resulting time is the sum of the creation of the keys plus the signing of the CSR by the CA's private key.

These operations are internal to the TPM because the objective is to measure the time difference between the different approaches. Indeed, the configuration of HLF for those measurements does not affect the performance since this configuration is an independent process to these cryptographic operations. The platform used for the measurements was the same as the one used in the proof of concept in the previous section: Raspberry Pi 4 Model B 8Gb, chipset Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC at 1.5GHz. The elements for executing the different algorithms involved in the proof of concept are the following:

- A TSS implementation, for TPM2.0, including a TPM command line interface, all integrated into TPM Tools Release 5.2 [69].
- TPM2 Access Broker and Resource Manager (TPM ABRM), Release 2.4.0 [70], for communication in the cryptographic operations between the TSS and physical or software TPM.
- OpenSSL V3 [71] for executing the cryptographic operations without using TPM.

In order to compare different scenarios, we have considered four approaches:

- Hardware TPM: Using TSS connecting via serial peripheral interface (SPI) to the RPI.
- Software TPM: Using the emulator of TPM developed by IBM [72].

- OpenSSL with Hardware TPM engine: Using cryptographic software like OpenSSL indicating specific engine (TPM2 TSS) in order to perform the operations.
- OpenSSL without engine: This approach only uses the OpenSSL library software.

Table 4.1 presents the results carried out for these four approaches. Note that from this table, the use of hardware TPM results in an increase in execution time in our proof of concept. Indeed, the results regarding the key generation, show that HW TPM is approximately 5 times slower than SW TPM, 30 times slower than OSSL and only 16 times slower than OSSL ENG approach. With regard to Sign operation, this is 5 times slower than SW TPM, 28 times slower than SW OSSL and 8 times slower than OSSL ENG version. In Verify operation, the HW TPM is 5 times slower than SW TPM, 17 times slower than OSSL and 9 times slower than OSSL ENG. Finally, in the commissioning mechanism, the HW TPM is 5 times slower than SW TPM, 50 times slower than OSSL approach and 8 times slower than OSSL ENG version.

These results are expected as the TPM to act as a hardware accelerator but as an element that offers a higher level of hardware and software security. The approach using Software TPM (SW TPM) is slower than software approaches using OpenSSL (5 and 3 times slower than OSSL and OSSL ENG for key generation, respectively). This is because the software TPM uses a socket for the communication between the TPM simulator driver and the TPM engine. In the case of operations using OpenSSL with TPM engine (OSSL ENG), the required time is much higher than using OpenSSL without TPM engine (OSSL) (almost 2 times slower in key generation and 6 times in commissioning), although it is less than Hardware TPM (HW TPM). This is due to the additional overhead generated by the communication between the OpenSSL stack and the TPM. It is clear that the fastest approach is to use software only (3.5 times faster than SW TPM in sign operation) as operations are not using the slow SPI link to retrieve data from the TPM and all the computing is made by the cores and the memory controllers which is much faster than the connection to the HSM. This software approach leads to it being the most widely used option in systems that do not implement any kind of security. The significant time overhead carried out when using a hardware TPM is a disadvantage in scenarios where high performance of real-time is required but the efficiency in the implementation of the smart contracts can help in hiding latencies if concurrent operations can be performed. So the selection of the HLF and the ability to implement the smart contracts is also key. Nevertheless, in systems where safety is a key factor, this time overhead is compensated by security features provided by a hardware TPM. In fact, TPMs were conceived to provide system robustness and a secure storage system. Its main features do not include the increasing of the processing speedup, which is reasonable since they are not designed for that purpose. HSMs in general provide co-processing when performing cryptographic operations, many servers use HSMs to speed up cryptographic operations of libraries such as SSL. What makes the

use of TPM specially suitable compared to other HSMs is that, in addition to offering the features of HSMs, it offers higher level mechanisms such as secure booting and remote attestation. The implementation of these mechanisms leads to TPMs following a standardization process and, hence, the software stack they implement is more robust. This standardizing element in TPMs is what makes it so promising in IoT systems as it is much easier to include in different environments, such as Blockchain networks.

TABLE 4.1: Execution time of the different approaches in milliseconds.

	HW TPM	**SW TPM**	**OSSL**	**OSSL ENG**
Create	0.5746	0.112	0.0197	0.0347
Sign	0.6030	0.1070	0.0212	0.0727
Verify	0.3379	0.0678	0.0195	0.0377
Commissioning	1.9934	0.3565	0.0384	0.251

In IIoT speedups are important but not losing messages and ensuring that the data feeding the smart contracts is legit a priority. Because this data will impact the big data pipelines of the industry. With the inclusion of a TPM module on the board of the IIoT node dedicated to data collection, it has to be considered that given the particular conditions of the design the gains of this approach are greater than the losses.

Regarding the timing differences of the hardware proposal versus the software emulated versions, it has to be considered that an ARM node running a conventional operating system has been used (it is not a native IIoT procedure) where it has to prioritize the tasks of attention to GPIO and the operating system tasks, in this scenario, communications via SPI are not particularly prioritized, so for example any block maintenance operation in the storage FFS modules would have higher priority than access to the SPI. This, in a specific IIoT node would probably not be so unbalanced. Still, the time differences are not an argument to consider this a non-viable solution. An IIoT node collects data over a period of time (it doesn't just collect a piece of data and immediately forward it to the Blockchain). This makes the time differences between the emulated and hardware versions negligible as data collection times overlap with TPM access times. According to Fig. 4.8, the actual situation of these nodes is such that:

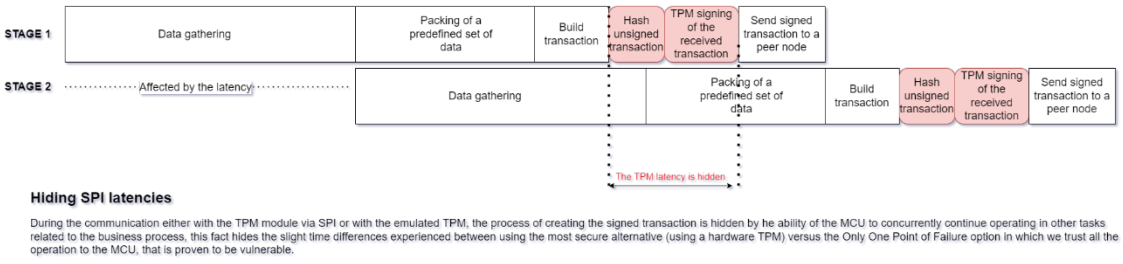

FIGURE 4.8: Comparison between TPM access time and IIoT data collection procedure.

Even so, as far as scalability is concerned, in this type of architectures we can find an important bottleneck in the loss of messages to be received by the transaction Orderer

node and in how the smart contracts to which the data incorporated in the transactions are directed have been implemented. Thus, for example, transactions with different timestamps can be computed concurrently [73]. The addition of a TPM can be efficiently hidden, as shown in Fig. 4.8 and that the existence of an increasing number of IIoT nodes equipped with this protection does not impact on the overall performance. However, it should be considered that in order to optimize the scalability of the whole architecture and its security, the nature of the transactions should be analyzed to exploit their parallelism to the maximum and be aware that the data processed by the smart contracts will be incorporated into a big data pipeline that will affect future business processes, so guaranteeing their security is crucial. Attacking the software repositories (and wallets) where keys and certificates are stored through smart contracts is nowadays a very fashionable attack vector. This is another reason why TPM should be included.

In this sense, Table 4.2 summarizes the four approaches analyzed in relation with the protection offered against three sets of attacks: Physical attacks [9], micro-architectural attacks [74] and software attacks [75].

TABLE 4.2: Security features of each approach.

	Physical Attacks	**Micro-architectural Attacks**	**Software Attacks**
Hardware TPM	✓	✓	✓
Software TPM	✗	✓	✓
OpenSSL	✗	✗	✗
OpenSSL with TPM engine	✓	✓	✓

4.4.6 Security analysis

Table 4.2 summarizes the different attacks against the different implementations studied for our proposal in the performance analysis. As can be seen, the use of a Hardware TPM is the approach that avoids the most kind of attacks. From a practical point of view, the proof of concept presented in this article is the basic resilience unit on which a IIoT network, consisting of different nodes implementing both a TPM and a Blockchain client, will be built.

The integration of the TPM with Blockchain technologies in networks of IoT nodes presents important security advantages. The main advantage of using a TPM in this type of architectures when compared to other technologies such as TEEs, is the ability to prevent physical attacks. Even so, within an IIoT environment, a network implementing this concept is exposed to other types of attacks:

- Denial of Service attacks [19]: As in our proposal the network has no central entity, attacks over specific nodes will not cause the network outage, thanks to decentralization provided by the Blockchain [53].
- Side channel attacks: HSMs and in particular TPMs avoid these kind of attacks offering tamper proof protection. Examples of these types of attacks are timing attacks [76] or fault induction techniques[77].
- Authentication attacks: Using multi-factor authenticated schemes implemented in the TPM and storing the credentials as private keys in hardware, make this type of attack useless in case an attacker wants to fraudulently access the Blockchain [78].
- Reverse engineering attacks: TPMs by offering tamper proof protection against invasive attacks, in which an attacker attempts to modify or alter the intrinsic functioning of the hardware, learning how it works or making it work as he wants, prevent such attacks [79].
- Replay attacks: This type of attack is compromised in Blockchain systems because for a transaction to be valid, it must be approved by the participants of the network. In the case of our proposal, the sending of erroneous data is detected in the verification process, even timestamp can be added to the data to mitigate this attack [80].
- Remote code execution attacks: This attack occurs when an attacker inserts code into the system to execute it at will. Thanks to remote attestation or trusted boot mechanisms, the TPM can check at runtime if malicious software is running [81].
- Sniffing: Also known as Man-in-the-Middle attacks. These attacks are solved thanks to encrypted communications over secure channels in which the keys are stored in the TPM. If an attacker is sniffing the channel, all he will see is the encrypted data and will not be able to decrypt it because the keys are securely stored in the TPM [82].
- Brute force attacks: Through trying combinations in the seed of a key generation, the attacker can find out a cryptographic key. Using TRNG implemented in the TPM, thanks to the high entropy of key generation, a brute force attack becomes impossible and very expensive [50].
- Impersonation attacks: An attacker can obtain the cryptographic keys of a user from the Blockchain and impersonate him, by storing the keys inside the TPM. These keys cannot be extracted, so an attacker will never be able to replace the user identity [52].
- Malware attack: Through a security breach or a peripheral, the attacker can introduce malware into the device. Mechanisms such as the trusted boot and the remote attestation prevent this kind of attacks [83].

Among all these types of attacks, software attacks are more common. Since the TPM is a standardized device by the TCG, it includes different mechanisms that a normal HSM does not support: it adds remote attestation and trusted boot to the device. In this case, as the RPI is an operating system supported platform both operations are supported. Trusted boot in the startup of the system helps to check the integrity of the device from the beginning. This integrity is checked also at run time using the remote attestation mechanism. This is a big step forward compared to HSMs, as it also provides mechanisms that check the state of the system during runtime. In addition, it is possible to establish TLS connections between devices, client and server. As a Blockchain is a decentralized network, this connection is performed between the nodes. The use of TPM reinforces the security of IoT devices and offers a high level of robustness to prevent these attacks through the mechanisms it implements

4.5 Conclusion

This paper proposes combining TPM and PB technologies such as HLF for building a trusted IoT node. Indeed, procedures for the interaction of TPM and HLF nodes have been presented. Furthermore, the proof of concept presented in this paper shows how, at a higher level, an IoT node interacts with the HLF Blockchain. The performance benchmark conducted in this paper sheds light on the possible uses of TPMs in the IoT world. As it has been shown, integration of TPM and Blockchain provides many advantages when interacting with each other: from the interaction at the operations level, such as signature and enrollment, to the applicability in industrial environments, adding the main characteristics that they bring as a technology.

Using this proof of concept as a main element, reliable and robust Blockchain networks can be built in which different attacks are mitigated. Future promising steps lie in the application of these two technologies together in a real IIoT environment, e.g. logistics or smart grids.

Indeed, Blockchain brings a wealth of benefits to industrial environments. Allows interoperability among enterprises, saving cost, eliminating bureaucracy, etc. These advantages added to the use of hardware secure elements makes the system fully robust, and enables secure end to end communication between different components through the Blockchain. The joint applicability of these two components in an industrial environment generates great added value and means that all monitored processes have a root of trust in the extraction of data and a root of immutability. The use of HSMs in Blockchain networks results in a very fruitful combination as on the one hand the keys that are stored and generated within the HSM cannot be extracted and, on the other hand, they add a layer of security when obtaining data from the devices located at the edge. In addition, using Hyperledger Fabric as a Blockchain network, improves latency

and scalability in transaction approval, creating a consortium where privacy and authentication are the main features which makes it suitable for industrial environments, creating an added value compared to traditional Blockchain networks, such as Ethereum or Bitcoin. All these considerations allow to augur a great future for these two technologies to go hand in hand thanks to the great advantages they bring together and their promising future in industrial environments. Finally, it should be noted that the proof of concept as well as the performance analysis testify to the seamless integration and future applicability in IIoT as the minimum security unit of a built network.

4.6 Acknowledgements

This work was supported in part by Infineon Technologies AG; in part by the European Union's Horizon 2020 Research and Innovation Program through the Cyber Security 4.0: Protecting the Industrial Internet of Things (C4IIoT) Project under Agreement 833828; and in part by FEDER/Junta de Andalucía-Consejería de Transformación Económica, Industria, Conocimiento y Universidades, under Project B-TIC-588-UGR20.

4.7 References

[1] Hugh Boyes, Bil Hallaq, Joe Cunningham, and Tim Watson. The industrial internet of things (iiot): An analysis framework. *Computers in industry*, 101:1–12, 2018.

[2] Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. Security and privacy challenges in industrial internet of things. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015.

[3] Sinem Coleri Ergen. Zigbee/ieee 802.15. 4 summary. *UC Berkeley, September*, 10(17):11, 2004.

[4] A Semtech. An1200. 22 lora modulation basics. *Semtech Application Note*, 2015.

[5] OASIS Standard. Mqtt version 3.1. 1. *URL http://docs. oasis-open. org/mqtt/mqtt/v3*, 1:29, 2014.

[6] Pedro Sanchez Munoz, Nam Tran, Brandon Craig, Behnam Dezfouli, and Yuhong Liu. Analyzing the resource utilization of aes encryption on iot devices. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1200–1207. IEEE, 2018.

[7] Jochen Mades, Gerd Ebelt, Boris Janjic, Frederik Lauer, Carl C Rheinländer, and Norbert Wehn. Tls-level security for low power industrial iot network infrastructures. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1720–1721. IEEE, 2020.

[8] Akshay Kumar Goel, Ajay Rose, Jayesh Gaur, and Bharat Bhushan. Attacks, countermeasures and security paradigms in iot. In *2019 2nd international conference on intelligent computing, instrumentation and control technologies (ICICICT)*, volume 1, pages 875–880. IEEE, 2019.

[9] George Loukas. *Cyber-physical attacks: A growing invisible threat*. Butterworth-Heinemann, 2015.

[10] Imran Bashir. *Mastering blockchain*. Packt Publishing Ltd, 2017.

[11] Ali Sunyaev. Distributed ledger technology. In *Internet Computing*, pages 265–299. Springer, 2020.

[12] David C Mills, Kathy Wang, Brendan Malone, Anjana Ravi, Jeffrey Marquardt, Anton I Badev, Timothy Brezinski, Linda Fahy, Kimberley Liao, Vanessa Kargenian, et al. Distributed ledger technology in payments, clearing, and settlement. 2016.

[13] Harish Natarajan, Solvej Krause, and Helen Gradstein. Distributed ledger technology and blockchain. 2017.

[14] Bin Cao, Yixin Li, Lei Zhang, Long Zhang, Shahid Mumtaz, Zhenyu Zhou, and Mugen Peng. When internet of things meets blockchain: Challenges in distributed consensus. *IEEE Network*, 33(6):133–139, 2019.

[15] Anjee Gorkhali, Ling Li, and Asim Shrestha. Blockchain: A literature review. *Journal of Management Analytics*, 7(3):321–343, 2020.

[16] Youssef El Hajj Shehadeh and Dieter Hogrefe. A survey on secret key generation mechanisms on the physical layer in wireless networks. *Security and Communication Networks*, 8(2):332–341, 2015.

[17] Keith Mayes and Konstantinos Markantonakis. Secure smart embedded devices, platforms and applications, 2014.

[18] Pinyaphat Tasatanattakool and Chian Techapanupreeda. Blockchain: Challenges and applications. In *2018 International Conference on Information Networking (ICOIN)*, pages 473–475, 2018.

[19] DENNIS GOH WEN QIN, UDPA NAMITHA SUJIT, LIMJUN JIE, and TEJASWI SINGH. Vulnerabilities and attacks on pki. *CS2107-Semester IV 2014-2015*, page 45.

[20] Tongtong Geng and Yueping Du. Applying the blockchain-based deep reinforcement consensus algorithm to the intelligent manufacturing model under internet of things. *The Journal of Supercomputing*, pages 1–23, 2022.

[21] Ru Huo, Shiqin Zeng, Zhihao Wang, Jiajia Shang, Wei Chen, Tao Huang, Shuo Wang, F Richard Yu, and Yunjie Liu. A comprehensive survey on blockchain in industrial internet of things: Motivations, research progresses, and future challenges. *IEEE Communications Surveys & Tutorials*, 2022.

[22] Gautam Srivastava, Jorge Crichigno, and Shalini Dhar. A light and secure healthcare blockchain for iot medical devices. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–5, 2019.

[23] Pradip Kumar Sharma, Neeraj Kumar, and Jong Hyuk Park. Blockchain-based distributed framework for automotive industry in a smart city. *IEEE Transactions on Industrial Informatics*, 15(7):4197–4205, 2018.

[24] Şeref Bülbül and Gökhan İnce. Blockchain-based framework for customer loyalty program. In *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, pages 342–346. IEEE, 2018.

[25] Chin-Ling Chen, Yong-Yuan Deng, Wei Weng, Ming Zhou, and Hongyu Sun. A blockchain-based intelligent anti-switch package in tracing logistics system. *The Journal of Supercomputing*, 77(7):7791–7832, 2021.

[26] Guido Perboli, Stefano Musso, and Mariangela Rosano. Blockchain in logistics and supply chain: A lean approach for designing real-world use cases. *Ieee Access*, 6:62018–62028, 2018.

[27] Se-Chang Oh, Min-Soo Kim, Yoon Park, Gyu-Tak Roh, and Chin-Woo Lee. Implementation of blockchain-based energy trading system. *Asia Pacific Journal of Innovation and Entrepreneurship*, 2017.

[28] Poonam Rani, Preeti Kaur, Vibha Jain, Jyoti Shokeen, and Sweety Nain. Blockchain-based iot enabled health monitoring system. *The Journal of Supercomputing*, pages 1–25, 2022.

[29] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[30] Lakshmi Siva Sankar, M Sindhu, and M Sethumadhavan. Survey of consensus protocols on blockchain applications. In *2017 4th international conference on advanced computing and communication systems (ICACCS)*, pages 1–5. IEEE, 2017.

[31] Gideon Greenspan et al. Multichain private blockchain-white paper. *URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf*, 85, 2015.

[32] JP Morgan. Quorum whitepaper. *New York: JP Morgan Chase*, 2016.

[33] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system

for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[34] Adam Everspaugh, Yan Zhai, Robert Jellinek, Thomas Ristenpart, and Michael Swift. Not-so-random numbers in virtualized linux and the whirlwind rng. In *2014 IEEE Symposium on Security and Privacy*, pages 559–574. IEEE, 2014.

[35] Emily Stark, Michael Hamburg, and Dan Boneh. Symmetric cryptography in javascript. In *2009 Annual Computer Security Applications Conference*, pages 373–381. IEEE, 2009.

[36] Miguel Herrero-Collantes and Juan Carlos Garcia-Escartin. Quantum random number generators. *Reviews of Modern Physics*, 89(1):015004, 2017.

[37] Najeh Kamoun, Lilian Bossuet, and Adel Ghazel. A masked correlated power noise generator use as a second order dpa countermeasure to secure hardware aes cipher. In *ICM 2011 proceeding*, pages 1–5. IEEE, 2011.

[38] Jen-Wei Lee, Ju-Hung Hsiao, Hsie-Chia Chang, and Chen-Yi Lee. An efficient dpa countermeasure with randomized montgomery operations for df-ecc processor. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 59(5):287–291, 2012.

[39] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted execution environment: what it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 57–64. IEEE, 2015.

[40] Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf. Trusted execution environments: properties, applications, and challenges. *IEEE Security & Privacy*, 18(2):56–60, 2020.

[41] Ross Anderson and Markus Kuhn. Tamper resistance-a cautionary note. In *Proceedings of the second Usenix workshop on electronic commerce*, volume 2, pages 1–11, 1996.

[42] Stathis Mavrovouniotis and Mick Ganley. Hardware security modules. In *Secure Smart Embedded Devices, Platforms and Applications*, pages 383–405. Springer, 2014.

[43] Sundeep Bajikar. Trusted platform module (tpm) based security on notebook pcs-white paper. *Mobile Platforms Group Intel Corporation*, 1:20, 2002.

[44] Trusted computing group. `https://trustedcomputinggroup.org`. Accessed: 2022-07-21.

[45] Trusted platform module library part 1: Architecture. `https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf`. Accessed: 2022-07-21.

[46] Trusted platform module library part 1: Structures. https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-2-Structures-01.38.pdf. Accessed: 2022-07-21.

[47] Chris J Mitchell, editor. *Trusted Computing*. IEE Press, 2005.

[48] Will Arthur, David Challener, and Kenneth Goldman. Platform configuration registers. In *A Practical Guide to TPM 2.0*, pages 151–161. Springer, 2015.

[49] Allan Tomlinson. Introduction to the tpm. In *Smart Cards, Tokens, Security and Applications*, pages 173–191. Springer, 2017.

[50] Will Arthur, David Challener, and Kenneth Goldman. *A practical guide to TPM 2.0: Using the new trusted platform module in the new age of security*. Springer Nature, 2015.

[51] Liang Gu, Xuhua Ding, Robert Huijie Deng, Bing Xie, and Hong Mei. Remote attestation on program execution. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, pages 11–20, 2008.

[52] Bryan Parno. The trusted platform module (tpm) and sealed storage. *TPM Documentation. June 21st*, 2007.

[53] Clemens Brunner, Fabian Knirsch, Andreas Unterweger, and Dominik Engel. A comparison of blockchain-based pki implementations. In *ICISSP*, pages 333–340, 2020.

[54] Andrew Miller. Permissioned and permissionless blockchains. *Blockchain for distributed systems security*, pages 193–204, 2019.

[55] Ijazul Haq and Olivier Muselemu Esuka. Blockchain technology in pharmaceutical industry to prevent counterfeit drugs. *International Journal of Computer Applications*, 180(25):8–12, 2018.

[56] Chao Lin, Debiao He, Xinyi Huang, Kim-Kwang Raymond Choo, and Athanasios V Vasilakos. Bsein: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. *Journal of network and computer applications*, 116:42–52, 2018.

[57] Zhetao Li, Jiawen Kang, Rong Yu, Dongdong Ye, Qingyong Deng, and Yan Zhang. Consortium blockchain for secure energy trading in industrial internet of things. *IEEE transactions on industrial informatics*, 14(8):3690–3700, 2017.

[58] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[59] Christian Cachin, Marko Vukolic Sorniotti, and Thomas Weigold. Blockchain, cryptography, and consensus. *IBM Res., Zürich, Switzerland, Tech. Rep*, 2016, 2016.

[60] Dan Boneh et al. Twenty years of attacks on the rsa cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999.

[61] Dindayal Mahto and Dilip Kumar Yadav. Rsa and ecc: a comparative analysis. *International journal of applied engineering research*, 12(19):9053–9061, 2017.

[62] Julio Lopez and Ricardo Dahab. An overview of elliptic curve cryptography. 2000.

[63] JW Bos, JA Halderman, N Heninger, J Moore, M Naehrig, and E Wustrow. Elliptic curve cryptography in practice in international conference on financial cryptography and data security, 2014.

[64] John D. Cook. Elliptic curve NIST P-384. `https://www.johndcook.com/blog/2019/05/11/elliptic-curve-p-384/`, May 2019. Accessed: 2022-8-23.

[65] Cameron F Kerry and Charles Romine Director. Fips pub 186-4 federal information processing standards publication digital signature standard (dss). 2013.

[66] David Jablon. Ieee p1363 standard specifications for public-key cryptography. In *CTO Phoenix Technologies Treasurer, IEEE P1363 NIST Key Management Workshop*, 2001.

[67] Magnus Nystrom and Burt Kaliski. Pkcs# 10: Certification request syntax specification version 1.7. Technical report, 2000.

[68] R Griffin and V Fenwick. Pkcs# 11 cryptographic token interface base specification version 2.40. *OASIS Open*, 2015.

[69] Release 5.2 2021-09-28 · tpm2-software/tpm2-tools. `https://github.com/tpm2-software/tpm2-tools/releases/tag/5.2`. Accessed: 2022-09-23.

[70] Release 2.4.0 · tpm2-software/tpm2-abrm. `https://github.com/tpm2-software/tpm2-abrmd/releases/tag/2.4.0`. Accessed: 2022-09-23.

[71] Openssl 3.0. `https://wiki.openssl.org/index.php/OpenSSL_3.0`. Accessed: 2022-09-23.

[72] IBM's software TPM 2.0. `https://sourceforge.net/projects/ibmswtpm2/`. Accessed: 2022-8-23.

[73] Parwat Singh Anjana, Sweta Kumari, Sathya Peri, Sachin Rathor, and Archit Somani. An efficient framework for optimistic concurrent execution of smart contracts. In *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 83–92. IEEE, 2019.

[74] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, 8(1):1–27, 2018.

[75] Jyoti Deogirikar and Amarsinh Vidhate. Security attacks in iot: A survey. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 32–37. IEEE, 2017.

[76] Sergei Skorobogatov. Physical attacks and tamper resistance. In *Introduction to Hardware Security and Trust*, pages 143–173. Springer, 2012.

[77] Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In *International Workshop on Security Protocols*, pages 125–136. Springer, 1997.

[78] Zhenlong Du, Xiaoli Li, and Kangkang Shen. Trusted firmware services based on tpm. In *International Conference on Trusted Systems*, pages 227–235. Springer, 2009.

[79] Piljoo Choi and Dong Kyue Kim. Design of security enhanced tpm chip against invasive physical attacks. In *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1787–1790. IEEE, 2012.

[80] Paritosh Ramanan, Dan Li, and Nagi Gebraeel. Blockchain-based decentralized replay attack detection for large-scale power systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[81] Hailun Tan, Gene Tsudik, and Sanjay Jha. Mtra: Multiple-tier remote attestation in iot networks. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2017.

[82] Jinchun Choi, Bohyun Ahn, Gomanth Bere, Seerin Ahmad, Homer Alan Mantooth, and Taesic Kim. Blockchain-based man-in-the-middle (mitm) attack detection for photovoltaic systems. In *2021 IEEE Design Methodologies Conference (DMC)*, pages 1–6, 2021.

[83] Dawei Li, Yingpeng Zhang, Jian Cui, Di Liu, Yu Sun, Zhenyu Guan, and Xu Wang. Remote audit scheme of embedded device software based on tpm. In *2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 61–66. IEEE, 2022.

5

Blockchain-Based Services Implemented in a Microservices Architecture Using a Trusted Platform Module Applied to Electric Vehicle Charging Stations

---

ANTONIO J. CABRERA-GUTIERREZ[1,2], ENCARNACION CASTILLO[2], ANTONIO ESCOBAR-MOLERO[1], JUAN CRUZ-COZAR[1,2], DIEGO P. MORALES[2], and LUIS PARRILLA[2].

1. Infineon Technologies AG, Am Campeon 1-15, 85579, Neubiberg, Bavaria, Germany
2. University of Granada, Department of Electronics and Computer Technology, Faculty of Sciences, Granada, 18071, Andalucia, Spain

Journal: MDPI Energies

- Received: 14 April 2023, Revised: 13 May 2023, Accepted: 21 May 2023, Published: 24 May 2023.
- DOI: 10.3390/en16114285
- Impact factor: Q3
- JCR Rank: 3.2, 78/115 Q3 in the category Energy Fuels

Abstract

Microservice architectures exploit container-based virtualized services, which rarely use hardware-based cryptography. A trusted platform module (TPM) offers a hardware root for trust in services that makes use of cryptographic operations. The virtualization of this hardware module offers high usability for other types of service that require TPM functionalities. This paper proposes the design of TPM virtualization in a container. To ensure integrity, different mechanisms, such as attestation and sealing, have been developed for the binaries and libraries stored in the container volumes. Through a REST API, the container offers the functionalities of a TPM, such as key generation and signing. To prevent unauthorized access to the container, this article proposes an authentication mechanism based on tokens issued by the Cognito Amazon Web Service. As a proof of concept and applicability in industry, a use case for electric vehicle charging stations using a microservice-based architecture is proposed. Using the EOS.IO blockchain to maintain a copy of the data, the virtualized TPM microservice provides the cryptographic operations necessary for blockchain transactions. Through a two-factor authentication mechanism, users can access the data. This scenario shows the potential of using blockchain technologies in microservice-based architectures, where microservices such as the virtualized TPM fill a security gap in these architectures.

5.1 Introduction

With the rise of Industrial Internet of Things (IIoT) environments, new computing and networking paradigms have emerged. This is due, among other things, to the fact that new security aspects have started to be considered. In traditional architectures, both client-server and publish-subscribe paradigms have been dominant [1, 2]. The problem with such architectures is that there are centralised entities, which play an important role within the network. A typical architecture example that implements a centralized client-server architecture is publish-subscribe Message Queue Telemetry Transport (MQTT) [3] protocol. In this model, some entities publish certain topics (publishers), and other entities that subscribe to those same topics obtain the published information (subscribers). Both publishers and subscribers are managed by a centralised entity called Broker. The Broker generates a point of vulnerability in the system, since Denial of Service (DoS) [4] attacks can have pernicious effects on the rest of the architecture. Being centralized entities, in which all requests are made against this entity, this type of attacks can collapse it and affect its availability, thus, as a consequence, the whole system is affected.

In addition, these protocols implement asymmetric cryptography to encrypt communications [5] and to prevent Man-in-the-Middle attacks [6]. This asymmetric cryptography is based on certificates issued by a Certification Authority (CA). This entity is part of what is known as a Public Key Infrastructure (PKI) [7]. PKIs are vulnerable, among other reasons, because of the centralisation of entities that play an important role, such

as CAs. Attacks on these entities can lead to the loss of credentials and the potential impersonation of a user by an attacker in the system [8].

These are the main reasons why technologies such as Blockchain have made a strong entry into the IIoT world in recent times. The Blockchain completely breaks the established client-server paradigm, replacing it with a peer-to-peer paradigm, decentralising the network and preventing the data it contains from being modifiable by third parties.

Blockchain offers trust and decentralization as its main characteristics. Based on Decentralized Ledger Technologies (DLT) [9], Blockchain distributes a database (ledger) among the entities, which belong to the network, improving the security by storing a copy of the database in each entity. Furthermore, Blockchain offers privacy since the hashes used by it make impossible to identify the data referring to a specific user and assures data protection from other parties by implementing access control policies. Immutability and traceability are also important features in Blockchain applications, that are reached by the transactions performed in the ledger by the different users. All the transactions are stored in the ledger and it can never be modified. Transactions change the content of the decentralized database, so they must be approved by the entities that make up the Blockchain according to the roles established within it. In this way any change is notified to the different entities keeping the integrity of the database [10]. Finally, it is worthy to mention the ability of some Blockchain applications to run use cases on this. These applications are executed through smart contracts that run on the ledger and perform actions predefined by the users who make up the Blockchain [11]. Thanks to this feature, Blockchain networks are starting to establish themselves in recent times in new fields of application as IIoT [12], healthcare [13], energy [14], business [15], financial [16], and others [17–19].

These benefits of Blockchain lie, in addition to decentralisation, in the use of cryptography in the transactions carried out. All these transactions are signed and subsequently verified in the Blockchain. All these cryptographic material (private keys, public keys, certificates) used to carry out the operations becomes of vital importance since it is where many of the properties of the Blockchain lie. If this cryptographic material is stolen or modified, it can have severe consequences for the integrity of the network [20].

One of the ways to effectively protect this cryptographic material, is the use of Hardware Security Modules (HSMs) [21], which securely stored the keys used in a Blockchain. In this way, as Figure 5.1 shows, a Blockchain client that has an HSM integrated can sign transactions using the private keys stored in the HSM securely, since these keys can never be extracted.

HSMs fulfill a very important factor such as the root of trust in systems where they are applied. In fact, HSMs are used in different fields as for example in PKIs [22], network protocols [23], database encryption [24], digital signatures [25] and cloud services [26].

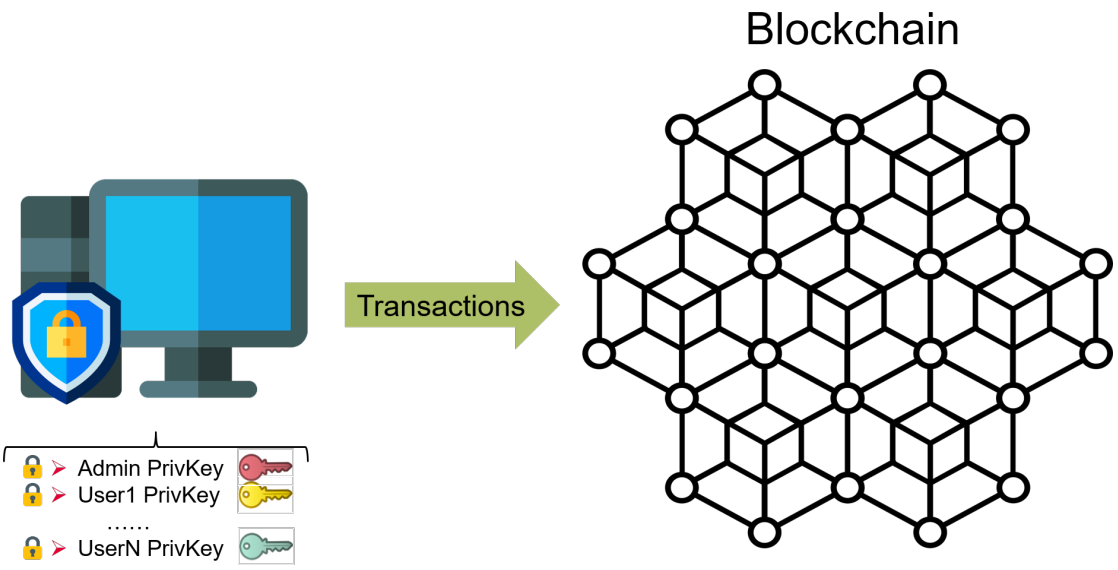

FIGURE 5.1: Blockchain with HSM.

In the case of cloud services, there are different applications running on the same hardware infrastructure (Infrastructure as a Service [IaaS]) where the HSMs are usually located. These HSMs are virtualized so that different software can make use of them, enabling providers to offer secure services to users, both at platform (Platform as a Service [PaaS]) and software level (Software as a Service [SaaS]) [27]. This virtualization of HSMs does not only occur in cloud services, but in any system where there is an orchestration of services that make use of the same resources. This is the case, for example, of an IoT gateway where different services are deployed. In fact, technologies such as edge [28] and fog computing [29] have made process virtualisation a more common concept.

This services can simply be processed in charge of a certain function in the system (e.g., maintain a database or run a web server). Virtual processes have given rise to microservices-based architectures [30], where the services are orchestrated in such a way that by interacting with each other, they are able to carry out correct functionality. This type of architecture offers much more efficient maintenance, deployment and reliability than traditional architectures [31].

This article proposes an HSM virtualization microservice which makes use of a special type of HSM such as the Trusted Platform Module (TPM). The virtualisation of the TPM is carried out through a Docker container which has a REST API to access it, offering high level functionality of it. In this container, different microservices delegate its cryptographic operations, attesting to the status of the container beforehand. These microservices are authenticated in the TPM container through Cognito Amazon Web Services (AWS) [32] in order to protect the API and establishing different permissions and roles when they use the TPM. Thus, in this architecture where different microservices have to make use of cryptography, they can access this container through

an authentication system that establishes different roles and permissions, which protect from unauthorized microservices. Finally, the article proposes an approach of this design to electrical vehicles (EVs) charging stations use case, where the data is stored in an EOS.IO Blockchain that makes use of the virtualized TPM. The main contribution over the state of the art is the virtualized TPM with the security mechanisms (attestation and private key sealing) provided for preserving the integrity and the security of the sensitive cryptographic material of the container as well as including a mechanism based on AWS Cognito for access control.

The rest of the article is organized as follows: Section 5.2 describes the work previously done regarding TPM virtualization. Section 5.3 presents the reason and the importance of use HSMs in combination of Blockchain. In Section 5.4, the proposed architecture design is described. Section 5.5 shows the application in EVs charging station use case. Finally, Section 5.6 summarizes the conclusion, emphasizing the benefits and applicability of this proposal.

5.2 Related Work

As a hardware resource, different cloud providers offer services where the TPM is virtualized. This virtualisation is known as virtual TPM (vTPM) and it is a software-based representation of a physical TPM [33]. vTPM concept is oriented to use it in virtual machines (VMs) and it provides the functionality of the physical TPM to other VMs. In order to get this functionality, vTPM is composed of a vTPM manager and different instances of vTPM. These instances run in the VMs and implement the full Trusted Computing Group (TCG) TPM2.0 specification. The vTPM manager creates the different instances and multiplex the request from the VMs to the vTPM instances. The communication between the VMs and the vTPM is done by using a driver model compound by a client side, running in the VM, and a server side, running in the vTPM. Figure 5.2 illustrates this architecture.

As it can be seen, this design requires implementing different drivers in each VM, as well as having one instance for each VM that uses the vTPM. This model does not fit properly in a microservices-based architecture, since this type of architecture rewards the modulation of the implemented microservices and the independence of one from the other [34]. In fact, this type of virtualisation, known as Hypervisor-based virtualization, differs significantly from container-based virtualization, which is mostly used in microservices architectures, providing much better performance than the aforementioned [35, 36]. Figure 5.3 shows the difference between both architectures.

In Hypervisor-based virtualization, a hypervisor (Virtual Machine Monitor) is running on the hardware enabling different VMs which share the same hardware resources, by this way, it is possible to emulate different Operating Systems (OSs) in the same platform.

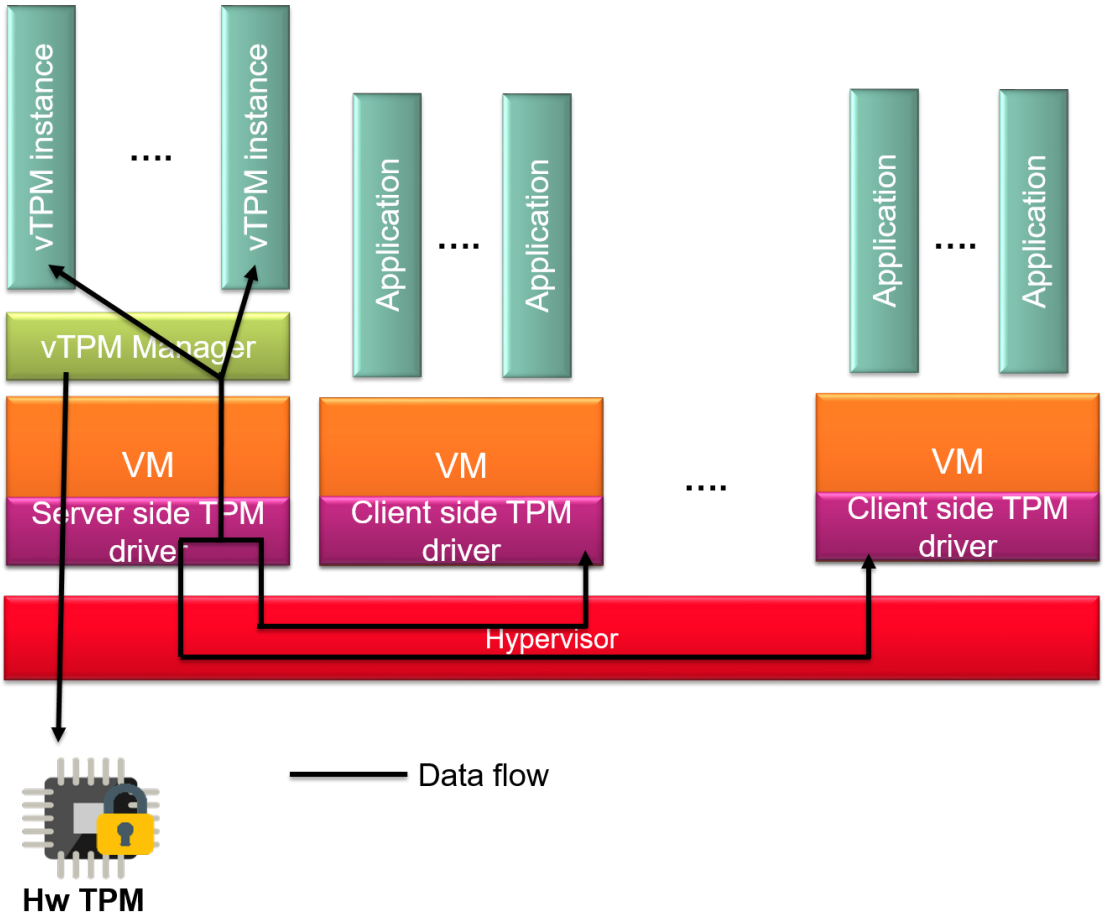

FIGURE 5.2: vTPM architecture.

In container-based virtualization, the container engine utilizes kernel features of the host OS to create isolated environments for applications, hence, applications build on this engine, share the hardware resources and the same OS, making them much lighter than those based on Hypervisor. Each of these technologies have their own advantages and disadvantages. Container-based virtualization has less performance overhead. As containers are more lightweight than VMs, it is faster to deploy and boot a container comparing to a guest OS. On the other hand, container-based virtualization has less flexibility in a way that it only hosts the same OS as the host platform is using [37].

However, as said before, container-based virtualisation brings many more benefits than hypervisor-based virtualisation for microservices architectures since they can deploy software applications as packets of modular and independent microservices, being much lighter, easier to maintain and reaching better performance [38]. Each microservice is a container which has its own programming language, data storage and communication mechanisms. Container-based virtualization have been a huge impact in microservice architectures, specially in IIoT applications which perform complex and new functionalities. In this kind of systems, a flexible and agile deployment as well as an efficiency in resources is needed. In addition, these systems tend to be very volatile, so good

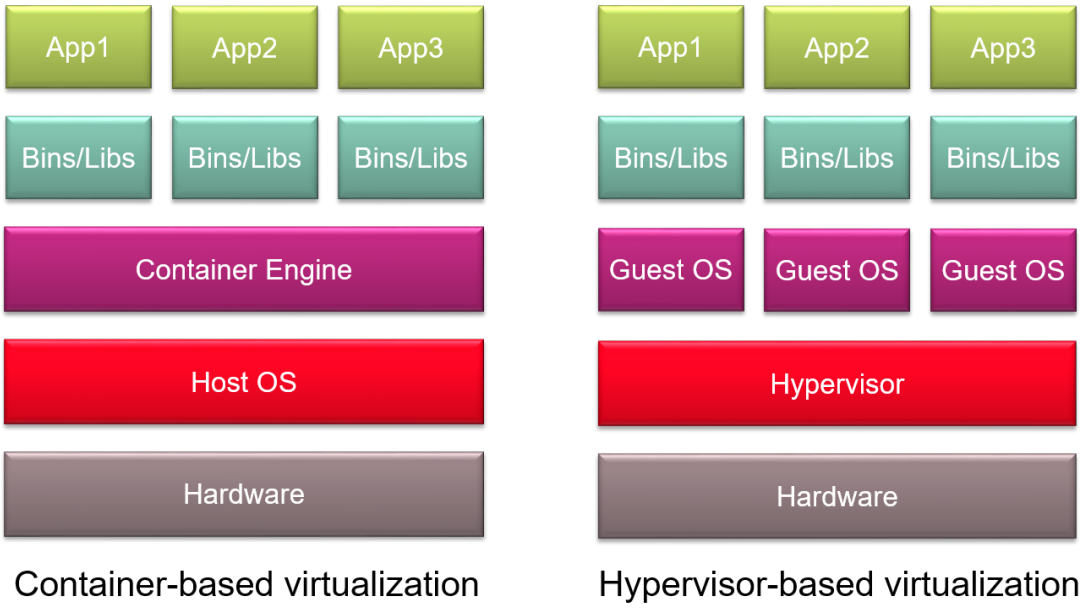

FIGURE 5.3: Hypervisor-based virtualization vs container-based virtualization.

scalability is also an important factor. In Figure 5.4, it is shown an example of microservice architecture for IoT. As it is shown, the different microservices interact between them in order to fulfil its purposes (collect data from different sensors, store them in a database and offer different analysis and visualisation techniques to external users). Each microservice is in charge of a task and communicates with the others independently. Microservices can range from managing a database to training machine learning algorithms.

However, it is not very common in the literature to find systems based on microservices that offer security aspects for IoT. In [39], the authors describe a security approach for developing microservices-based IoT systems. This approach combines microservices' patterns, APIs, distribution of microservices, and access control policies. Attribute-Based Encryption (ABE) scheme is proposed in order to deploy identity management, authentication and policy controller in microservices.

The work proposed in [40], presents an IoT framework which incorporates security microservices as role management, authentication, access control and identify governance.

In [41], the authors propose a smart surveillance system based on microservices architecture and Blockchain technology. The communication between the mircroservices is encrypted using Advanced Encryption Standard (AES) [42] and Rivest-Shamir-Adleman (RSA) [43] cryptography algorithms. The Blockchain network is used to ensure the decentralization of the data exchanged by the different microservices, as well as to introduce access policies to this data through smart contracts.

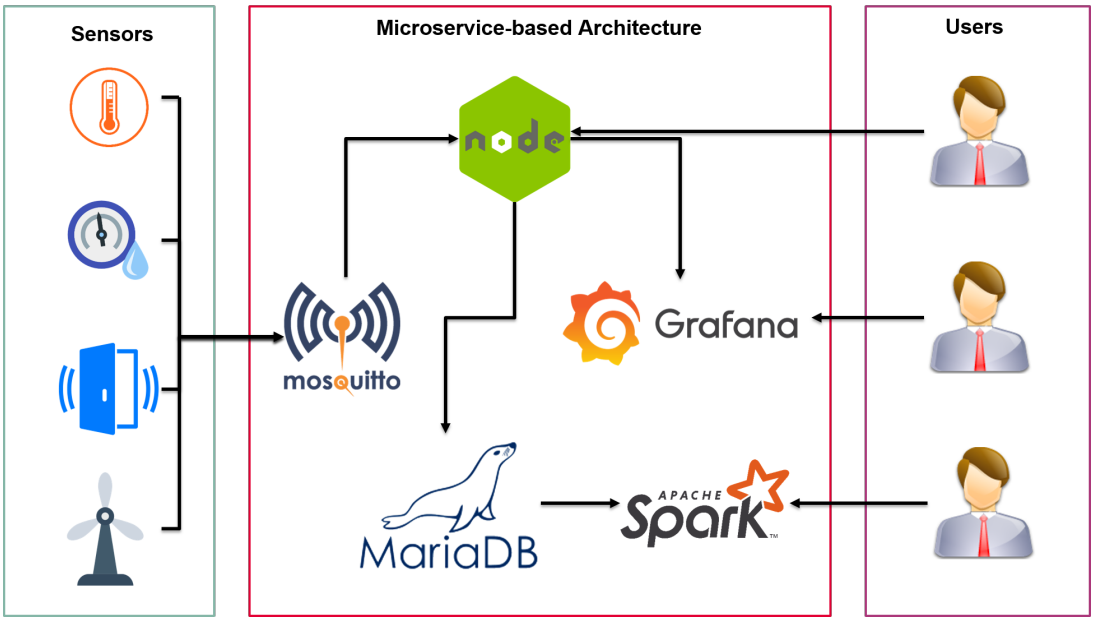

FIGURE 5.4: Example of microservice-based architecture.

The work described in [44] proposes different Blockchain services deployed on edge nodes. These services are related to Blockchain operations as execution of consensus algorithms, generation of new blocks and performing mining tasks.

In [45], the authors propose a secure edge computing management based on microservices, with a gateway that is responsible of manage devices, data, users and configuration securely. The gateway incorporates an API used to filter all incoming request in order to protect the microservices running on the edge.

In [46], the security solution is a machine learning based approach. In this approach, the microservices-based model identifies strange behaviors, observes communication packets and intercepts malicious traffic which potentially may damage to the system.

The work proposed in [47] describes a solution using a traffic monitor which runs on the IoT nodes, routers, etc. in the network. The traffic monitor intercepts the packages between the services and identify the communication patterns of each data transmission. By this, the model can classify the communication as normal or anomalous.

All of the above works offer security between microservices, or prevent malicious attacks on them through different mechanisms or policies. However, none of them offer security at the cryptographic level, i.e., they do not provide any microservice in which different applications use cryptographic keys, signatures, or certificate storage through a microservice that manages this. In this article, a microservice is proposed that can be used by others to provide keys, perform signatures and other cryptographic operations related to a TPM.

5.3 Trusted Platform Modules in Combination with Microservices-Based Architectures

In microservices architectures, hardware security has not been addressed so far. HSMs in this type of architectures is absent in most of the literature. However, providing a root of trust in the system built on top of HSMs is of vital importance.

HSMs offer a hardware root of trust since they generate cryptographic keys using high-entropy random number generators, and protect them through secure storage (keys stored internally are never extracted). They offer cryptographic algorithms implemented in hardware for encryption/decpryption or signature generation and verification [48].

Within HSMs, there is a special subgroup formed by TPMs, which differ from other HSMs in that the methods implemented in them are standardized by the TCG [49] as well as offering higher level mechanisms than a common HSM [50]. TPMs fit better in virtualized architectures since they offer mechanisms such as secure boot [51] or remote attestation [52] that ensure system integrity at startup and during the execution period, detecting any software modification. Moreover, it is ideal for use on systems with OS support, which offer virtualization support as it has already discussed in the previous section introducing vTPM [33].

The problem, as mentioned above, is that vTPM does not fit perfectly into the philosophy of the microservices' architecture, as these are implemented in container-based virtualization. To solve this problem, a few approaches are presented in the literature [53–55], which have been analyzed at the time of the design of the proposal presented in this article.

Figure 5.5 presents a first approach to combine TPMs with microservices [33]. In this, each microservice, making use of the TPM, has a vTPM in the container. Therefore, in this approach, the size of the containers may grow considerably, resulting more heavy and less flexible than other proposals.

To make this approach lighter, one solution adopted is to move the vTPM to the operating system level [54]. By this way, the TPM is available to different containers by assign a vTPM instance to a new container. The container manager asks the host OS kernel to create a new vTPM instance and then assigns it to the new container. Figure 5.6 shows this scheme. In this case, the kernel must be trusted because an attack on it can lead to information leakage by the vTPM. To ensure a trusted kernel, the solution taken must extend the root of trust from the TPM to the kernel, through mechanisms such as remote attestation or secure boot. Thus, if the kernel is considered secure, all the vTPMs attached to it are secure as well [56]. The problem with this solution, applied to microservices architectures, is the lack of flexibility when deploying this system on any platform. As is well known, microservices systems create virtualized environments that

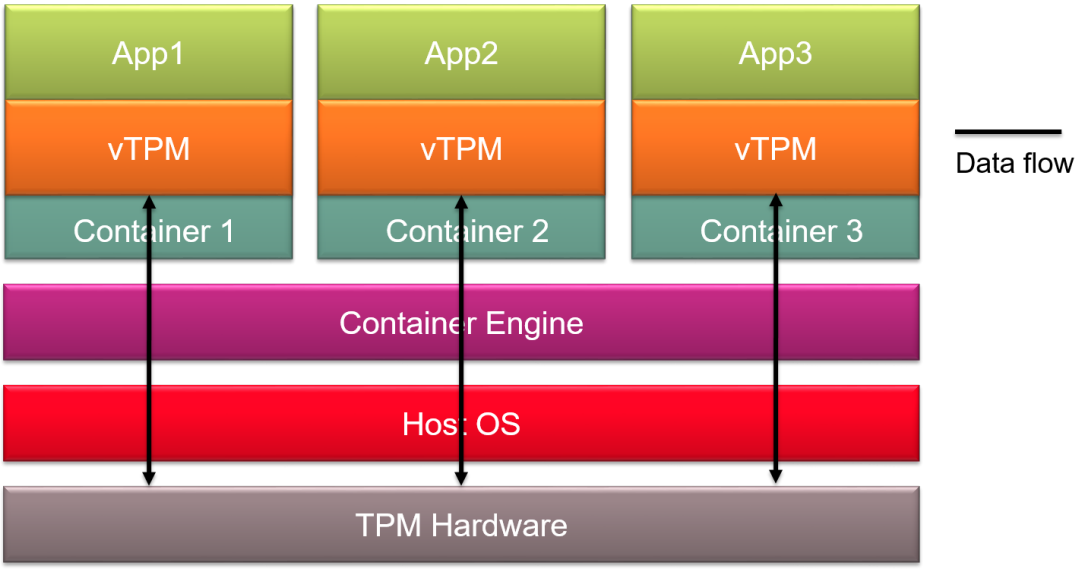

FIGURE 5.5: Approach with vTPM inside each container.

are independent of the OS and the hardware platform implemented in underlying layers. If this proposed solution uses kernel modules/resources to different microservices, these microservices will be more difficult to port and deploy on different OS architectures.

The other approach [55] that we will consider consists on dedicating results in dedicating a container exclusively to the vTPM, i.e., creating an exclusive microservice in which different containers that require the use of the functionalities provided by the vTPM must communicate with it. In this way, an appropriate flexibility and portability related microservices' system is achieved. Figure 5.7 illustrates this solution. The difficulty of this solution is to ensure the reliability of the container carrying the vTPM. In this sense, it is necessary to make use of the attestation mechanisms offered by the TPM, both in the deployment of the container and during its use. It must also be protected from malicious microservices that try to access the vTPM with an authentication system. These aspects will be discussed in detail in the next section.

Note that in this proposal any microservice's system deploying vTPM microservice will be able to realize the functionalities offered by the TPM. In this sense, many microservices implement cryptographic security in microservices-based systems: from an MQTT server to a Blockchain client, going through database services, using the cryptographic capabilities to sign transactions, exchange encrypted messages using symmetric cryptography or encrypt and decrypt files.

However, if it digs deeper into the technical aspects, not all microservices could be compatible with this vTPM microservice. TPMs are able to generate secure keys within them because they incorporate high-entropy True Random Number Generator (TRNGs). They are based on an external physical phenomena, making it almost impossible for a key generated with a TRNG to be replicable. These generated keys are used for

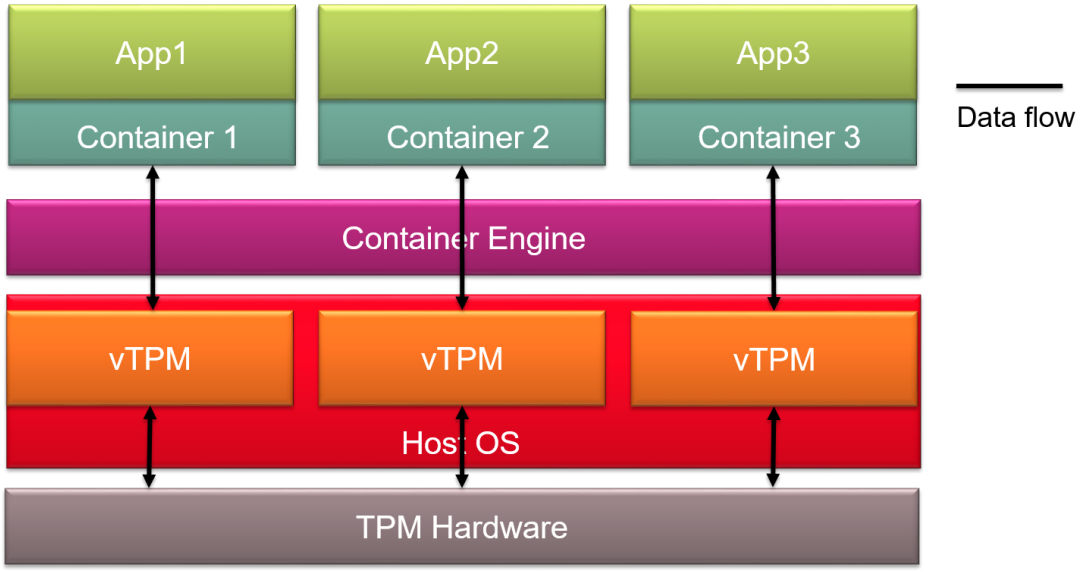

FIGURE 5.6: Approach with vTPM in the OS kernel.

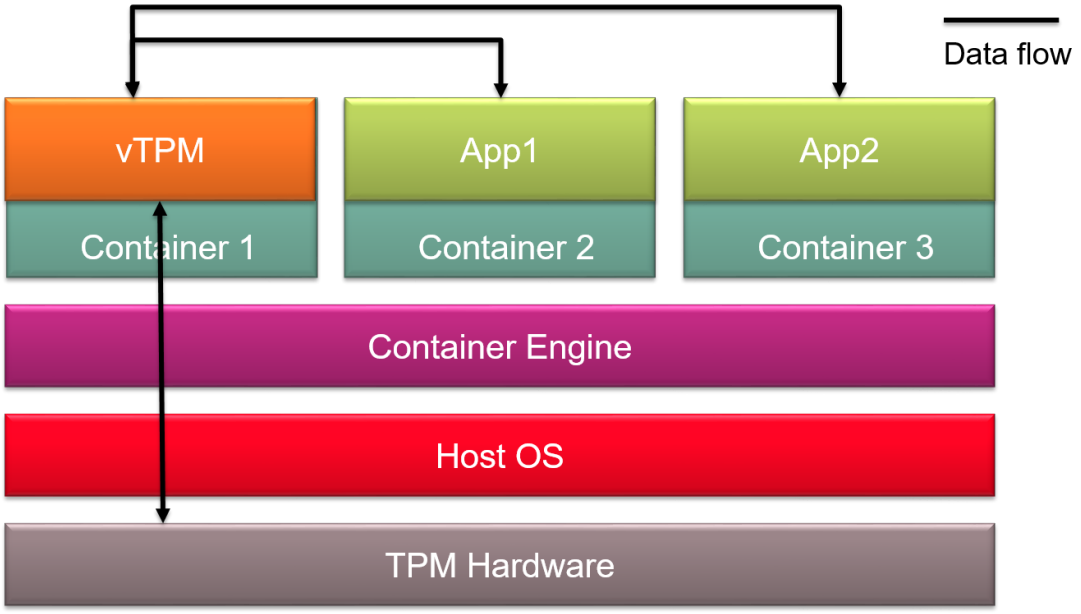

FIGURE 5.7: Approach with vTPM in a dedicated container.

implementing public-key cryptosystems based on elliptic curves. There are a wide variety of Elliptic Cryptographic Curves (ECCs) usually supported by TPMs as NIST standard [57] prime256v1, or secp256k1 which is the one used in Bitcoin [58] or Ethereum [59]. Since the cryptographic operations used in microservices are performed in the TPM, these two technologies must support the same cryptographic curves and algorithms, making this a mandatory property when integrating these two elements.

Instead, this type of solution allows for the possibility of abstracting the software-level functionality of the TPM. The software implemented by the microservice that makes use of the TPM is independent and does not have to comply with any standard to make use

of the TPM. This is a differentiating element of the traditional use of TPM by external applications, since there are software standards that are used to manage a TPM as if it were a hardware token, as the case of PKCS11 [60]. This standard defines an API in order to interact with the TPM by external applications.

In this way, complete container-based virtualized systems can be built making use of vTPM microservice, being flexible and easily portable to any type of platform. Based on this approach, we will proceed to explain the architecture design proposed in this article.

5.4 Architecture Design

The design proposed in this article is based on the third approach discussed above, in which the TPM is virtualized in a container and the other containers, whether they are on the same hardware platform or on a different one, can access it through an interface and a communications protocol. In this way, an architecture based on microservices can make use of cryptographic material from the TPM, creating a root of trust in this type of architectures. The proposed design improves the third approach by adding important security features as private key sealing, attestation and access control to the vTPM, at the cost of a negligible loss of performance.

Architectures that implement Blockchain microservices use cryptography with great frequency, since these applications base their reliability on the cryptographic operations they perform. This is the case of transactions, messages sent to the Blockchain and signed by the sender. This signature is done through a private key that is generally generated and stored in software. Through the use of this microservice, the keys are generated inside the TPM and all cryptographic operations are performed inside it. Thus, the external microservice that makes use of it will only receive the results of these cryptographic operations. In this way, the keys are securely stored internally as it prevents physical attacks such as probing [61] or side channel attacks, timing attacks [62] or fault induction techniques [63]. By this way and thanks to the TPM, hardware attacks are mitigated, although, the architecture proposed in this article involves different aspects that must be addressed in terms of security.

The vTPM container must offer security mechanisms that prevent an attacker from modifying the container and injecting malicious code inside it, and the microservices that make use of the vTPM container must follow access policies that establish different permissions and authorisations when performing functionalities within the vTPM. For example, a Blockchain client may have permission to perform cryptographic signatures in the TPM, but may not have permission to generate cryptographic keys. It is logical that this operation is available to other microservices with more authority, such as a Blockchain administrator.

In order to solve this, the vTPM microservice implements a secure attestation functionality by the microservices that make requests, so that when a microservice is going to make a request to it, the microservice first performs an attestation to check that the vTPM is correct and it has not been modified. In addition to this, a secure deployment must be performed to prevent any process from interfering in the deployment and causing any failure in it, or any malicious code injection during the deployment. In relation to the protection of the microservice from the use of other microservices without permissions, an authentication service based on Cognito AWS [32] is used, which, through different policies, establishes different access rights to other microservices that have been previously registered with it.

In the following sections these two aspects will be detailed, but first, it will proceed to explain the proposed virtualization of the vTPM in a container.

5.4.1 Virtualization

The vTPM built in this proposal is inside a container and must be communicated with others through extenal APIs. Typically, TPMs are the hardware base of different applications which are built on top of it. These applications use different APIs implemented in the middleware in order to use the TPM functionalities. As it was said before, PKCS11 is one of these APIs.

The software used in the TPM is standardized by the TCG and it is composed of diverse software: from the lower functionality in order to manage the TPM resources to the higher level functionalities through Python libraries. Figure 5.8 illustrates the different software components enabling the use of the TPM, which are:

- TPM2 Access Broker and Resource Management Daemon (ABRMD) [64]: This software module is compound by two different parts: the Access Broker and the Resource Management. The Access Broker manages synchronization between the processes that use TPM simultaneously, while the Resource Manager manages the TPM context in a similar way to the OS memory manager. TPM generally have very limited memory, thus TPM data (objects, sessions and sequences) need to be swapped from the TPM to the memory to allow TPM commands to be executed. ABRMD works on the Unix hardware device representation of the TPM (*/dev/tpm0* and */dev/tpmrm0*).
- TPM2 Software Stack (TSS) [65]: The TSS is compound of the following layers, from the highest level of abstraction to the lowest: Feature API (FAPI), Enhanced System API (ESAPI), System API (SAPI) and TPM Command Transmission Interface (TCTI). All these APIs offer the TPM functionalities to applications written on it. These APIs are implemented in C and C++ which means a low

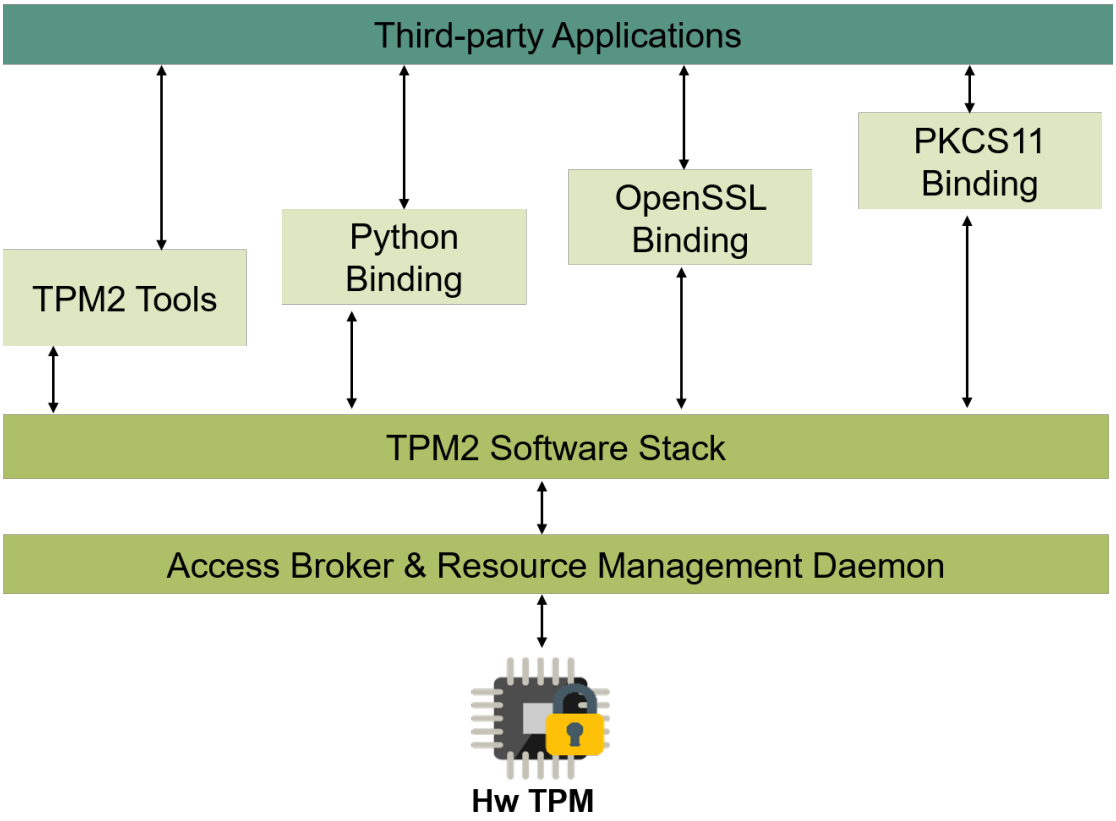

FIGURE 5.8: TPM software.

portability and compatibility with external applications, that is why there are different libraries that are built on top of the TSS.

- TPM2 Tools [66]: It provides a set of bash binaries which interacts with the different functionalities of the TSS.
- TPM2 Python [67]: It provides a Python API in order to access to the TSS.
- TPM2 OpenSSL [68, 69]: It implements a provider and an engine for OpenSSL v3.0, making the TPM accessible for OpenSSL API and command line tools.
- TPM2 PKCS11 [70]: It offers a PKCS11 based API in order to get the TPM accessible as a hardware token.

All of this software is essential to make the TPM work, however, many of the bindings mentioned are redundant to each other and can be bypassed for certain applications. For example, an application built in Python will use the libraries provided in this language for the use of the TPM, and it will not be necessary to include the command line tools. To avoid incompatibilities between applications built on top of this type of software and to make it more flexible, it has been created a REST API on top of the software described above. This API only uses TPM2-Tools and OpenSSL bindings in order to interact with the TPM. In this way, the virtualized software inside the container is

reduced and the TPM can be accessed by external microservices that make use of the cryptographic operations offered by the TPM through the REST API. Figure 5.9 shows the complete vTPM architecture, where the main component is the Docker container which has all the software modules required.

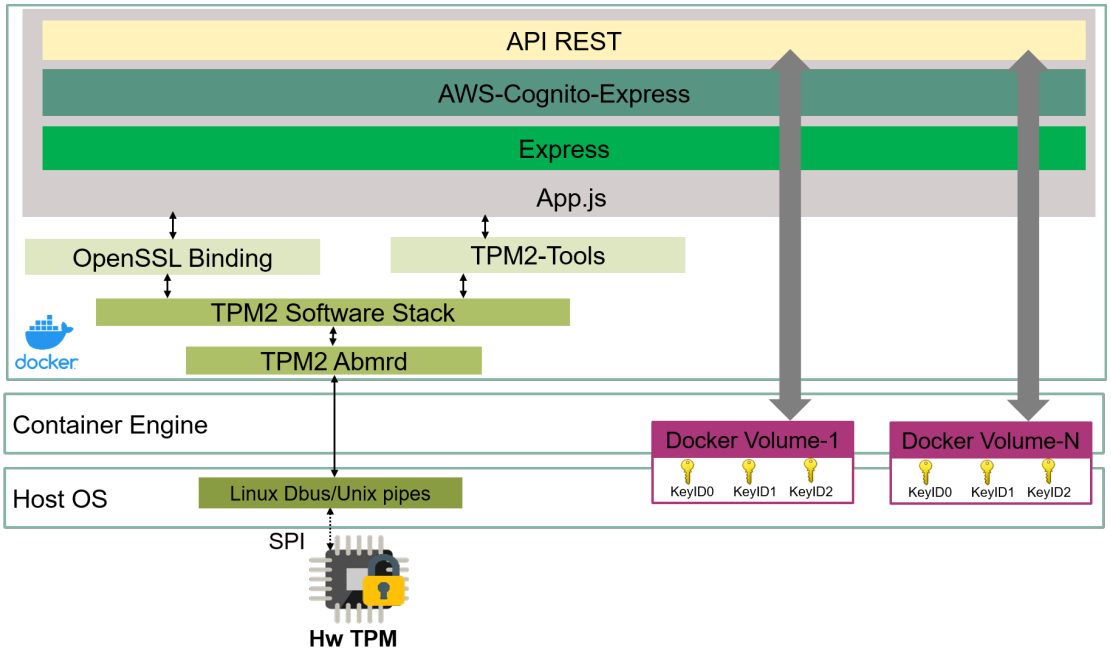

FIGURE 5.9: vTPM architecture.

The REST API is an express server built with NodeJS. This REST API implements an authentication mechanism through Cognito AWS which will be discussed in Section 5.4.3. This container uses volumes to store the identifiers of the private keys created through the API, depending on different permissions and roles established through Cognito AWS, the keys will be created in different volumes. These identifiers are used to access the private keys that are stored inside the TPM. The application built in NodeJS is in charge of calling the underlying software (TPM2-Tools and OpenSSL bindings) which in turn communicates with the TPM through the software stack discussed above.

Therefore, the REST API is responsible for providing functionality to the container from the point of view of an external application, so that this API exposes different entry points in which each of them performs one or more operations within the TPM. The entries available in the API are the following:

- Create key: This operation generates a private key. The name of the key is generated randomly following a Binary Large Object (BLOB) format. This BLOB is given as a return message of the operation. The private key is stored internally in the TPM, but his identifier named with the BLOB is stored in a volume.

- Signature generation: The signature process made inside the TPM is done through this entry point. The input parameter are the BLOB identifying the private key which will made the signature and the hash which will be signed.
- Export public key: This entry point returns the public key associated to a private key identified with a BLOB.
- Signature verification: This operation verifies the signature using the public key. The entry point receives the signature, the hash and the BLOB identifying the private key. The return value is a boolean indicating the success of the verification.
- Get random: Returns a random value using the TRNG into the TPM. The input parameter is the length in bytes of the random value requested.
- Encrypt: This operation encrypts using the private or public key, depending on the algorithm chosen in the input parameters, a string given. The return value is the encrypted string.
- Decrypt: This operation does the inverse process than the previous operation, decrypting a given string in the input parameters.
- Hash: It computes a hash, following the algorithm given in the REST function input parameters. The return value is the hash computed in hexadecimal format.

In addition to these operations, there are other mechanisms which perform attestation and integrity validation of the microservice since all these vTPM functionalities would be useless if an attacker intervenes and accesses the microservice, modifying it and redirecting this information to another microservice and impersonating it or even creating fakes BLOBs for the keys. These mechanisms will be explained in the next section.

5.4.2 Attestation and Data Sealing of the vTPM Microservice

To ensure the correct operation of the deployed microservice, different checks of this microservice status must be performed. These checks are the most important security aspects in the container, since all the trustworthiness of the operations executed in the vTPM microservice may collapse due to an attack on the integrity of the container and therefore, on the TPM functionalities.

The underlying idea is to ensure that the microservice running inside the container always provides the results it should, ensuring the immutability and integrity of the container, and to transmit that confidence to the client microservices that use that microservice. The first step to achieve this is to get a secure deployment on the platform where you want to make use of this microservice. To do this, it must be ensured that the underlying hardware and software does not contain any foreign element that could

be exploited in the future as a security flaw. In this sense, part of the responsibility lies on the hardware resource provider (PaaS), since if there is malicious middleware or a backdoor in any hardware component, all the hardware security mechanisms that are built in higher layers will be useless.

The next step is the deployment and the integration in the platform (Continuous Integration and Continuous Development (CI/CD)). In this regard, there are tools that check the security of the container and perform tests on it. They also check the containers for known vulnerabilities in real time and warn about them so that they can be fixed immediately [71]. It could even be deployed through a secure script, in which it is launched through checks of an external TPM that verifies that the script is correct and has not been modified by any external process. In addition, a check of the system on which the script will be launched can be performed, as well as the different binaries used for the deployment. However, these mechanisms are external to the inner workings of the vTPM microservice and are therefore not the subject of this article's in-depth study; each vendor is responsible for offering its own mechanisms, as they exist for traditional containers with microservices that are not necessarily dedicated to security.

What is of interest in this article, and where the fundamentals of the security-related operation of this container are based, is in the checking of the vTPM during the execution of the container. During this time, the possible lack of integrity of this container is unavoidable, some OS or even remote process can get access to this container through a backdoor or a software bug. The container software cannot be shielded since, as a container, it is running in memory zones within a hardware platform where there is an OS and different containers or external processes. For this reason, the information that the vTPM must be restricted and can only be released if an integrity check is performed on the container previously, and this check is valid. In contrast to the container data, this information can be shielded because it is stored inside the vTPM. In this sense, when an external microservice makes a request to the vTPM microservice, it must "unlock" this information so that it can be sent to the external microservice.

TPM2.0 provides mechanisms to carry out this functionality. Moreover, during the execution time, it can perform measurements of different elements of the operating system and the rest of the software to check the status of these components. These measurements are stored in special TPM registers called Platform Configuration Registers (PCRs) [51]. Depending on the value of these registers, the TPM will unlock the information required by the client microservice (the generation of a signature or the use of a private key). The PCRs are used both for the request that a microservice makes when using the TPM and unseal the data contained in the TPM measuring the state of the container during execution. The first is known as attestation of the container by the TPM, and the second is known as sealing process.

5.4.2.1 Container Attestation

The objective of the PCRs is to measure the software state on the platform. TPM2.0 has 24 PCRs, each of them has a predefined value at reset state. These PCRs values change after perform different measurement over the OS kernel and file system. The operation which made these values change is called extension and the resulting value is obtained by concatenating the incoming data-digest with the current value at the PCR index, hashing the concatenated data and replacing the PCR index value.

To check the state of the container, several of the PCRs are used by extending them with hashes from different files in the container, e.g., binaries installed inside the container or from the REST API code.

Figure 5.10 shows the process that follows a client container to perform the attestation and subsequent verification. If this verification is satisfactory, the client container will perform operations on this container that it deems appropriate, such as the creation of keys or obtaining random numbers. In this way, making sure that the state of the vTPM container is correct, the client knows that the operations performed on it are reliable.

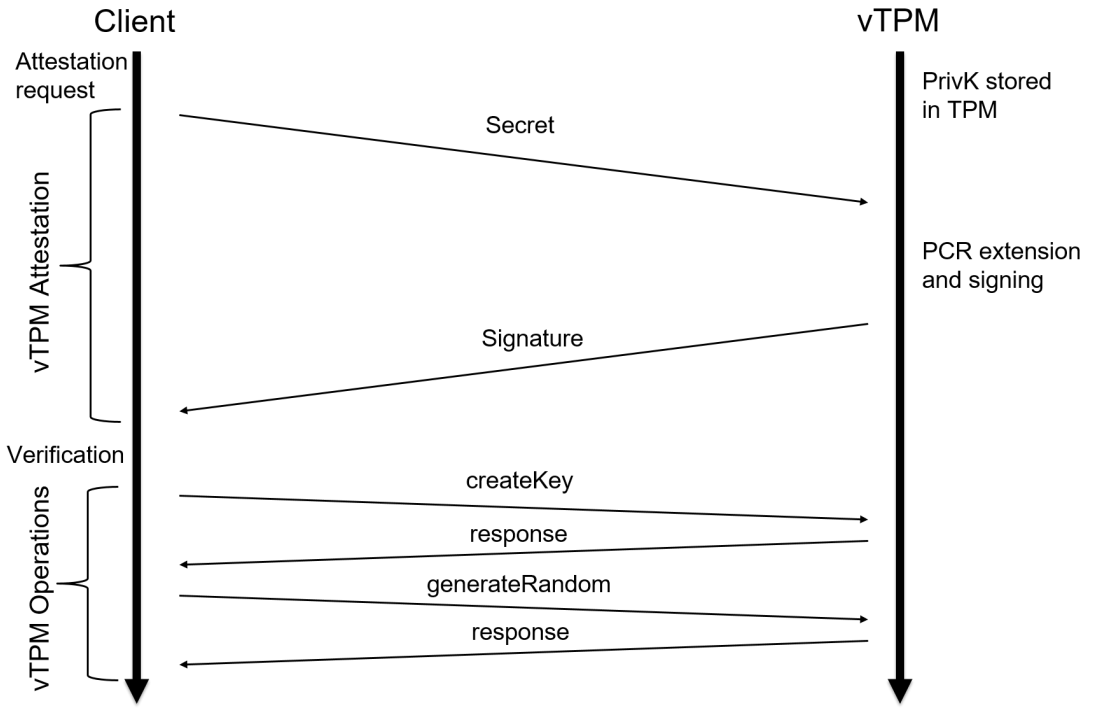

FIGURE 5.10: Container attestation mechanism.

Going deeper into the mechanism of attestation, the client sends a secret to the vTPM, being this secret randomly generated in the client and avoiding reply attacks [72]. The vTPM performs the hashes to different files in the container and extends the PCR values chosen by the vTPM which are indicted in the vTPM policies. All the PCR values used

in the operation are hashed following one combination predefined by the TPM and results in one single data hash. This final hash is signed using an Attestation Identity Key (AIK) and the result is sent back to the container client, which verifies the signature and the hash. This verification process depends on the TPM provider which owns the TPM and knows the state in which the hash returned by the vTPM is correct. In addition, it has a certificate issued from the AIK that serves to verify the signature. Therefore, when the microservice is deployed, the TPM provider must be responsible for providing to the container client both the attestation validation certificate and the correct hash for verification. This information may be made available through a microservice offered by the TPM provider or sent to client containers that make use of the vTPM microservice. PCRs are protected with different mechanisms offers internally by the TPM, implementing policies and session authorization [50] in order to do some modifications on these registers. Therefore, an attacker who wants to maliciously manipulate PCRs must first have obtained these credentials or authorization rights which are owner by the TPM itself.

Thus, this mechanism requires a measurement of the container software prior to deployment. During this measurement, the TPM provider assumes that the container has not been modified, and its functionality is uncorrupted. When it is redeployed within a microservices' architecture, all microservices that make use of it should have the original measurement available for verification. Thus, every time a microservice wants to access the vTPM microservice, it will know that the container status it is in the one certified by the provider. Same mechanism can be performed by the owner of the hardware platform in order to check the integrity of the hardware providing a proof that the platform is not corrupted and the microservices can be deploy securely.

In conclusion, PCRs values are reported in a signed attestation quote, permitting a external container to determine the vTPM container software's trust state through a valid authenticity and integrity proof offered by the TPM provider.

5.4.2.2 Private Key Sealing

When a microservice uses the vTPM, the container stores the private keys it generates in a container volume. Actually, they are not the private keys, since these are stored inside the physical TPM, but they are files that work as identifiers of these keys, accessing to these, it is possible to make use of the private key stored inside the TPM. These files are identified with a BLOB that is randomly generated at key creation. When a key is created, the API returns the BLOB to the client microservice.

However, storing this type of material in a container volume is dangerous, since a container volume is a file system that is shared with the host OS. If it is compromised by some attackers, they can access these files and obtain these key identifiers. This is

the reason why key identifiers stored within a volume must be protected. To solve this problem, this paper proposes to make use of the sealing process that incorporates the TPM. In this process, a file is sealed using the state of certain PCRs of the TPM. Once sealed, the file cannot be used until it is unsealed again by setting the PCRs indicated to the value they had at the time of sealing. The Figure 5.11 shows the seal and unseal processes.

The client sends a secret in the request to generate a private key. This secret, which is actually an 8-digit password, is in charge of establishing the combination of PCRs to be extended and in what order to perform the sealing. The secret is decoded, and the 8 digits are divided into four two-digit numbers, which are transformed into module 24 to correspond to one of the PCRs of the TPM. The order of extension is established, starting with the most significant number of the secret and ending with the least significant.

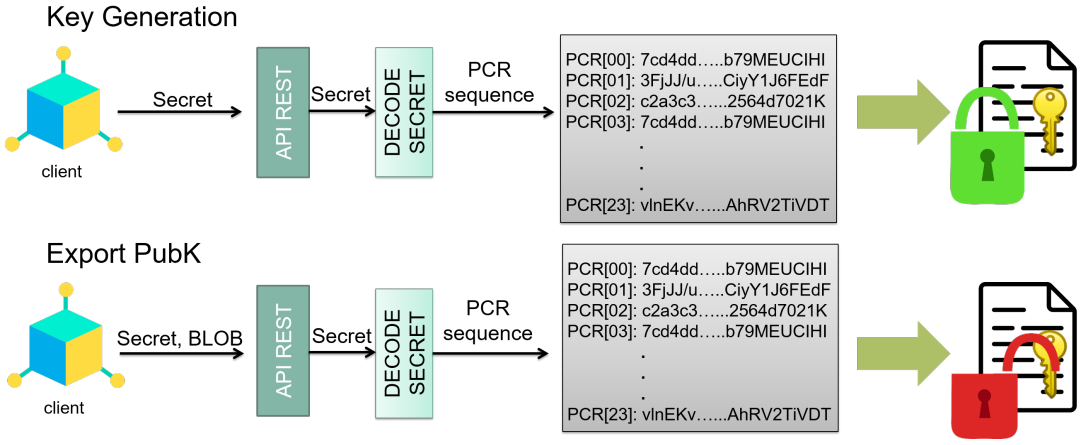

FIGURE 5.11: Seal and unseal processes.

In this way, the private key is created and sealed with the contents of the PCRs. The content of these PCRs reflects the state of the container, thus, the key is sealed with a password imposed by the client and with the state of the container when the key is created.

The generation of the key operation creates a BLOB that is returned to the client, so that, in subsequent API calls that make use of that private key, that key can be referenced. For example, as in Figure 5.11 is illustrated, in the export public key operation the client must send the BLOB of the private key, which be used to export the public key, and the secret in order to unseal the private key. If the secret results in a correct extension of the PCRs and the PCRs contains the same value of the container when it generated the key, the private key will be unseal and the operation will be performed exporting the corresponding public key.

Thus, through the PCRs of the TPM, the integrity of the deployed container and its content can be ensured, on the one hand through measures that are verified by the provider, and on the other hand, through passwords that serve to unlock sensitive material stored

inside the container. In conclusion, for each operation that the client wants to perform in the vTPM microservice, to ensure the reliability of the data obtained, first, the client must attest that the integrity is the one insured by the provider and second, the private keys must be unlocked for use through a password that the client knows and that only works if the state of the container is the same as when that key was created.

Once the integrity of the container has been ensured during the use through the mechanisms explained above, it only remains to explain how to protect it from unauthorized microservices.

5.4.3 Cognito Amazon Web Service

Cognito AWS is a service that provides access management, authorization and authentication of entities. Through Users Pools, the registered containers are managed according to established policies where different roles and permissions can be set. Thanks to this, a secure and restricted access to the REST API exposed in the vTPM container is guaranteed.

The mechanism to ensure this is shown in Figure 5.12. When a container wants to access the vTPM microservice, it must first register within an User Pool. This process is done using a username and password, and then, the container will be in the User Pool and will have the rights to authenticate later.

When authenticating, the container must use the username and password used in the registration. Once this is done, Cognito AWS returns a token, which is used from now, until the time of use expires, in all operations performed by the container on the vTPM microservice. When a request is made with this token, the vTPM microservice checks if it is valid and what permissions this token has associated with it (e.g., permission to create keys or to sign). Permissions are established through a serie of associated attributes indicating the operations that the token is potentially capable of performing, such as signing, extracting a public key or generating a random number.

If the token used in the request is invalid, either because it has expired or it is a fake token, the transaction will not be carried out. As well as if the client container wants to carry out an operation for which it does not have the required permissions.

Each User Pool has different permissions that are given according to the operations that the microservice performs. That is, if a Blockchain client microservice needs to make use of signing, the User Pool where it registers will only allow this operation, however, a Blockchain administrator microservice must be able to generate private keys for clients, in this case, the User Pool where it registers will allow the creation of keys in the vTPM microservice. It must be emphasised that the microservices registered in the User Pool

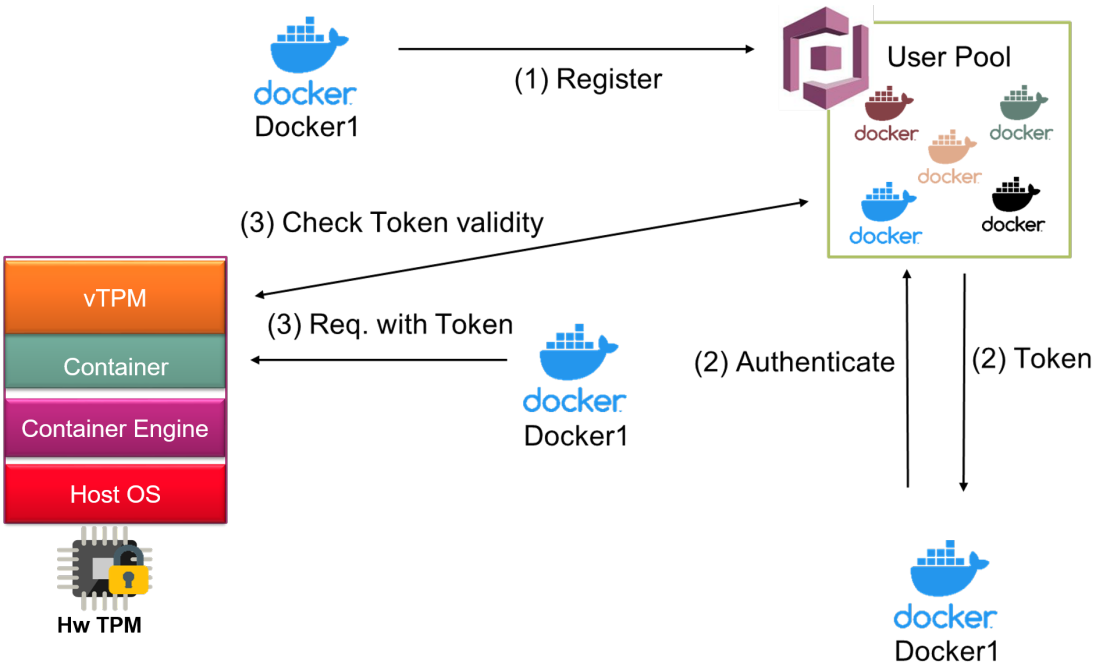

FIGURE 5.12: Authentication process with Cognito AWS.

must be done through the Pool administrator, who knows in advance that the registered microservice is reliable.

In addition, as each User Pool has specific permissions associated with it, it makes sense that the clients that are registered in that User Pool share the same volume in the container. Thus, a group of containers that perform Blockchain operations, for example, share the same keys generated in the same volume, as these keys are used for the same purpose. It would not make sense, for example, for these keys to be shared with an MQTT client microservice. So in this way, independent microservices do not share the same space, avoiding possible mixing of private keys in the volumes.

In this way, through Cognito AWS, the API and the functionalities offered are protected against containers that are not authorised to use it, thus protecting it against possible attacks by malicious containers. In addition, setting permissions between different containers and separating the volumes used between different User Pools, provides better modularises the functionalities offered, by separating the keys generated by the container between microservices.

Thanks to Cognito AWS and the attestation and sealing mechanisms of the private keys, a robust and secure container is achieved against possible attacks on the system where it is deployed, ensuring correct functionality and detecting any unauthorised intrusion into the system.

Finally, a point to take in consideration is that the use of this type of architecture introduces a loss of performance related to the security mechanisms and the fact of to

using a virtualization platform: any virtualized system introduces a loss of performance at runtime, it is always faster to run a code closer to the hardware than to do it in higher abstraction layers, introducing overhead in the system calls through the different layers of virtualization. The only way to mitigate this as much as possible is to optimize the code running in the container.

5.5 Electrical Vehicles Charging Stations Use Case

The potential offered by this microservice concept is easily exploitable in different use cases, furthermore, taking advantage of the cryptographic operations which the container exposes through the API is ideal for use with Blockchain applications, which make intensive use of cryptographic mechanisms: key generation, signing or signature verification. There are many implementations of Blockchain in the energy field [73–75], showing the potential of this technology.

One of these cases is EVs charging stations. EVs are becoming more common and charging stations are an expanding infrastructure, especially in urban areas. Thus, the connection between electric vehicles and Blockchain technology is promising [76]. The microservices-based architecture proposed in this article for this case offers a charging station geolocation service, where all its usage is collected in a decentralised Blockchain database preserving privacy and offering data integrity in the records. This system is shown in Figure 5.13. On the one hand, the information related to the EVs charging stations is stored in the Blockchain through a NodeJS API. The data collected are the geographical position of the charging station, the energy supplier of the station, the power it supplies, the type of vehicle that can be connected and the kind of energy source from which the offered power is generated. This kind of data is stored in the registration process of the charging stations and is immutable until a Blockchain transaction modifies it, e.g., if the type of energy changes from a renewable to a non-renewable source. Through a charging station finder microservice, the user enters the search criteria which are the distance to the station, the estimated charging time, the money available and the type of energy source. If the charging station ceases to be operational, it will itself send a revocation certificate to the Blockchain to register until it is operational again by re-registering within the network. User privacy is guaranteed through the Blockchain, as it provides anonymity to the transactions made. Only the user identifier appears in these transactions, which is a cryptographic identifier based on a cryptographic hash function. With this information, the system offers the user different charging points that suit his preferences. In addition, the system has a history record microservice, which allows the user to query all the energy charges performed in the system. This microservice provides private information such as a record of bank transactions, money spent, time spent and charging stations used, so it must be protected through an authentication system based on Cognito AWS and a hardware token. Both this microservice and the Blockchain

client make use of this Cognito-based authentication system. As explained in Section 5.4.2, the Blockchain client must be authenticated in Cognito AWS in order to use the vTPM microservice. The Blockchain client is in charge of sending information to the Blockchain as well as querying it. Both the information related to the EVs charging stations and the information related to the charges made by the users are sent through the Blockchain client to the Blockchain network in the form of transactions previously signed by the vTPM microservice. This Blockchain client microservice is an EOS.IO node.

In this way, it is possible to track all energy usage of a user, as well as all vehicles that have been charged at a given charging station. These records are immutable thanks to the Blockchain. The system is deployed on a platform that has a TPM which is virtualized through the vTPM microservice following the mechanisms discussed in Section 5.4.2.

In the following sections the two main parts of this systems will be explained: the Blockchain client that makes use of the vTPM and the hardware token authentication system with Cognito AWS (two-factor authentication).

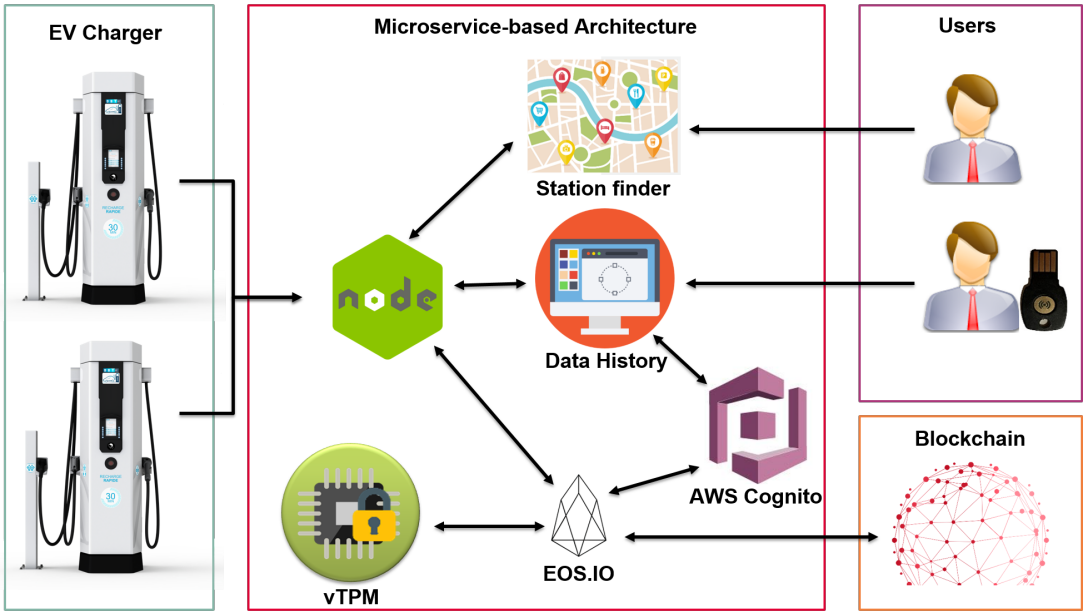

FIGURE 5.13: EVs charging microservice-based architecture porposed.

5.5.1 EOS.IO

EOS.IO [77] is a Blockchain designed for deploy and execute decentralized applications or so-called smart contracts. This Blockchain is designed to prioritize smart contracts performance. EOS.IO uses a variation of Proof-of-stake algorithm known as Delegated Proof-of-Stake (DPoS). This consensus algorithm regulates all the processes related to staking tokens, voting, vote decay, vote recording, producer ranking, and inflation pay.

Smart contracts are written in C++, being the code compiled and then deployed in an EOS.IO virtual machine.

EOS.IO client consists of following three modules:

- Cleos: It is the command line tool that connects to the API exposed by Nodeos and works to manage the wallet, account, keys, transactions and smart contracts.
- Nodeos: It works as the central daemon that manages the EOS.IO network and can be configured as a node to produce blocks, which are be able to sign blocks.
- Keosd: It is in charge of storing and generating the keys.

EOS.IO is the Blockchain microservice in charge of performing the necessary operations with the vTPM microservice and it is ideal for applications where good performance is required, being this the main reason why it has been chosen for this use case. In this scenario, instead of using Keosd as the key manager, the vTPM API is used.

It should be noted that in the architecture depicted in Figure 5.13, only one EOS.IO node is shown, but additional EOS.IO nodes may appear in the use case that make use of the vTPM. It has been represented in this way for simplicity. Therefore, this EOS.IO node (or nodes) is externally connected to other nodes in a distributed network. This network can scale considerably in the system once deployed, typically, Blockchain networks tend to have scalability issues once deployed in potentially large systems. In this case, EOS.IO introduces different types of scalability (horizontal and vertical) as well as data access. As for vertical scalability, EOS.IO improves performance by introducing enhancements to the Nodeos componet. To improve horizontal scalability, EOS.IO leverages the use of various abstraction layers for smart contracts. These abstraction layers allow multiple types of Blockchain systems to easily use these smart contracts in a efficient way. Regarding data access scalability is implemented to authenticate transactions that query and read history and state data, minimizing the exchange of data between entities.

EOS.IO node authenticates with Cognito AWS into the vTPM microservice for use cryptographic operations as key generation and transaction signature. The NodeJS microservice provides the operation that EOS.IO will perform. If the operation is a transaction, the NodeJS script will provide the data as well. The transactions add data to the ledger, which can be either related to EV charging station information (geographical location, supplier, etc.) or to an energy charge (power supplied, money spent, bank details, etc.). The transaction is made with the corresponding data and signed by the vTPM microservice, before that, the Blockchain client must check the status of the vTPM microservice, making an attestation and checking the values obtained with those provided by the cloud operator. In addition, in order to perform the signature,

it must unseal the private key, identified with the BLOB obtained in its generation. The transaction is sent to the Blockchain and it is approved by different nodes in the Blockchain following the consensus algorithm implemented in EOS.IO. When using the geographic location of the charging station or the payment history microservice, they generate a query to the Blockchain through the Blockchain client. These microservices never write to the ledger, so operations performed on the Blockchain are simply queries.

In this way, through one or more Blockchain nodes running on the system that make use of the vTPM microservice, data is stored on the Blockchain, maintaining an immutable record of it and being able to build smart contracts that execute some processing logic. Also note that in order to be able to rely on these smart contracts, they must be verified minimizing the risk of faults and bugs [78].

5.5.2 Two-Factor Authentication

The proposed mechanism for user authentication follows the same scheme as the one used in Section 5.4.3. This mechanism allows users who need to authenticate to the system to do so through a two-factor authentication system that is more secure than a traditional one. In this sense, the user carries a USB hardware token where the user's private keys are securely stored. The authentication is done through Cognito AWS.

In this scenario, Users Pools have different roles assigned to the users, e.g., for clients with special permissions as network administrators. In addition, the Users Pools assign roles depending on the functionalities allowed to the users. In this way, Users Pools have a similar functionality to the microservices authentication use case seen in Section 5.4.3.

The registration and authentication mechanism is shown in Figure 5.14. The user registers and logs into the system through a web interface. At registration, the user enters his/her name and password in the web form. This name and password must be previously stored in the User Pool in order to allow the registration to the user, this step is performed prior to registration through a different means (e.g., by e-mail to the system administrator). When the registration process starts, the web interface generates a challenge, which is sent to the hardware token. In the hardware token, the public and private key pair is created. Using the private key, the challenge is signed and sent to the Cognito User Pool. The pool stores the user's attributes (name, email, etc.), user ID and public key.

At this point, the user can log into the system. To do so, in this process, the user enters his/her username and password. These data are checked in Cognito AWS to see if the user has already been registered, if so, Cognito AWS sends a challenge to the hardware token. This challenge is signed with the private key on the token and sent to Cognito AWS which, through the previously stored public key, is able to verify the signature and check that the authenticating user is really who claims to be. In this process, both

challenge creation and challenge verification operations are performed through lambda functions deployed in Cognito AWS.

After this process, Cognito AWS sends a token to the user that will be used to authenticate to the web interface until the token expires. Through this token, the user can access the microservices that the token allows, being this token associated to the user and the User Pool that contains it. A token issued to a user belonging to the administrators' User Pool will have more access rights to special functionalities (e.g., only administrators can create keys) than a normal user.

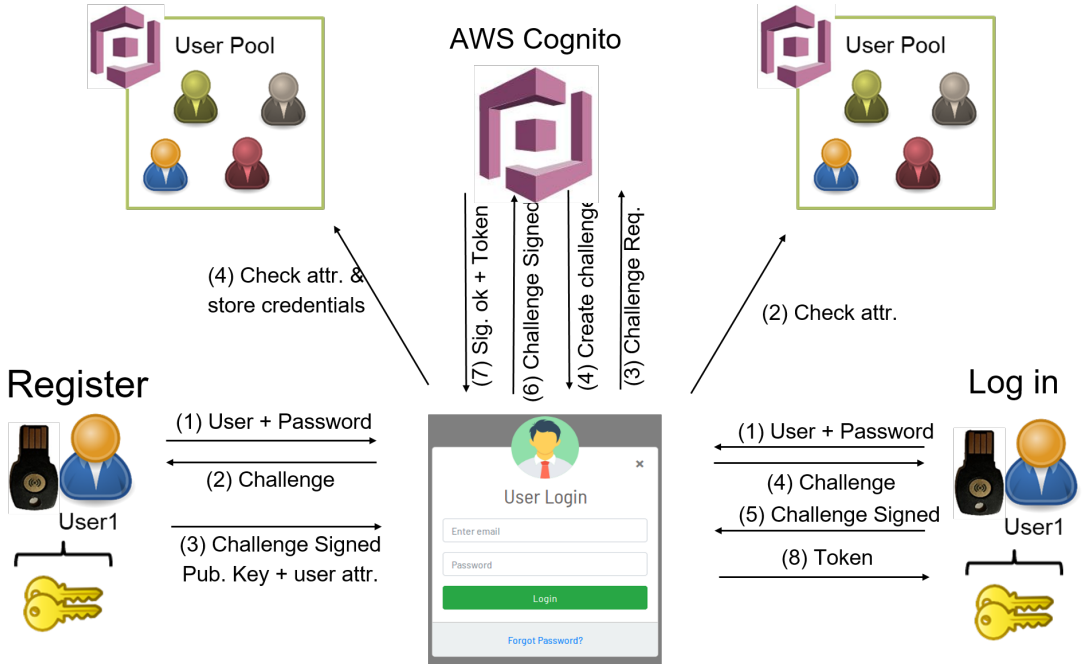

FIGURE 5.14: User authentication process with Cognito AWS.

In this way, a secure and robust authentication system is achieved, with different user roles, protecting the system from unauthorised access and keeping credentials securely in hardware.

5.6 Conclusions

Blockchain technologies are gaining prominence and are increasingly being used. In particular, in microservices-based architectures, the use of Blockchain microservices opens up a wide range of possibilities for this type of architectures. The use of cryptography that powers the Blockchain is an important aspect to address and offering microservices of this type improving the security of microservices-based architectures.

The use of microservices that offer cryptography through a virtualisation of a hardware security module such as the TPM, allows the hardware root of trust to be extended to other applications. This is why it is important to create a virtualisation of this type

of hardware resource. Through the virtualisation proposed in this article, the vTPM microservice is capable of offering the functionalities that a TPM provides to other microservices such as Blockchain clients. Thanks to this microservice, the security of the rest of the microservices is increased as the cryptographic operations are delegated to the vTPM.

On the other hand, this microservice must be secure and maintain integrity in both deployment and use. Attestation mechanisms ensure that the container is not modified by external entities. This mechanism, through special TPM registers such as PCRs, performs different measurements inside the container, checking that the libraries, binaries and the rest of the source code have not been modified since the initial state, which is supposed to be a secure state. In addition, private key identifiers are stored in container volumes that are protected through a sealing mechanism. Using this mechanism ensures the integrity of these identifiers and its usability by owners, as these identifiers can only be unlocked through a password or secret that unseals the identifier using the TPM's PCR. In addition to these integrity mechanisms, this container has been protected with an authentication system based on Cognito AWS, which establishes permissions and roles for other containers to access the vTPM microservice. This protects against unauthorised access by untrusted third party microservices.

As a potential offered by this proposal, a use case applied to EVs charging stations has been described where all the information related to the payments is sent to an EOS.IO Blockchain which makes use of the vTPM by signing the transactions and generating the keys used. The information stored on the Blockchain is immutable and all recorded information are accessed by users who use a two-factor authentication mechanism to gain access to sensitive information.

The future of Blockchain microservices lies in the integration and virtualisation of hardware security elements. This proposal shows a potential application in the energy sector such as EVs charging stations and helps the union of these technologies to reach a higher level of maturity, without forgetting the hardware security that is required.

5.7 Acknowledgements

This work was supported by Infineon Technologies AG. Additional funding was provided by the program "Digitalisierung der Energiewende", sponsored by the "Bundesministeriums für Wirtschaft und Energie" (BMWi) in the frame of "7. Energieforschungsprogramms der Bundesregierung", through the funding project "Trusted Blockchains fur das offene, intelligente Energienetz der Zukunft (tbiEnergy)", under FKZ 03EI6029D. In addition, this work was partially funded from European Union's Horizon Europe research and innovation program through the funding project "Cognitive edge-cloud with serverless computing" (EDGELESS) under grant agreement number 101092950. The work

has also been funded by FEDER/Junta de Andalucia-Consejeria de Transformacion Economica, Industria, Conocimiento y Universidades under Project B-TIC-588-UGR20.

5.8 References

[1] Saritha Saritha and V Sarasvathi. A study on application layer protocols used in iot. In *2017 International Conference on Circuits, Controls, and Communications (CCUBE)*, pages 155–159. IEEE, 2017.

[2] Karthikeyan Ponnusamy and Narendran Rajagopalan. Internet of things: a survey on iot protocol standards. *Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2016, Volume 2*, pages 651–663, 2018.

[3] OASIS Standard. Mqtt version 3.1. 1. *URL http://docs. oasis-open. org/mqtt/mqtt/v3*, 1:29, 2014.

[4] Fu Chen, Yujia Huo, Jianming Zhu, and Dan Fan. A review on the study on mqtt security challenge. In *2020 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 128–133. IEEE, 2020.

[5] M Sundarrajan, AE Narayanan, and V Srithar. Securing the mqtt protocol using enhanced cryptographic techniques in iot surroundings. In *Journal of Physics: Conference Series*, volume 1767, page 012055. IOP Publishing, 2021.

[6] Henry Chiyang Wong. *Man-in-the-Middle Attacks on MQTT based IoT networks*. PhD thesis, Missouri University of Science and Technology, 2022.

[7] Joel Weise. Public key infrastructure overview. *Sun BluePrints OnLine, August*, pages 1–27, 2001.

[8] Maurizio Talamo, Franco Arcieri, Andrea Dimitri, and Christian H Schunck. A blockchain based pki validation system based on rare events management. *Future Internet*, 12(2):40, 2020.

[9] Ali Sunyaev and Ali Sunyaev. Distributed ledger technology. *Internet computing: Principles of distributed systems and emerging internet-based technologies*, pages 265–299, 2020.

[10] Karim Sultan, Umar Ruhi, and Rubina Lakhani. Conceptualizing blockchains: Characteristics & applications. *arXiv preprint arXiv:1806.03693*, 2018.

[11] Shafaq Naheed Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, and Anoud Bani-Hani. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-peer Networking and Applications*, 14:2901–2925, 2021.

[12] Wattana Viriyasitavat, Tharwon Anuphaptrirong, and Danupol Hoonsopon. When blockchain meets internet of things: Characteristics, challenges, and business opportunities. *Journal of industrial information integration*, 15:21–28, 2019.

[13] Cornelius C Agbo, Qusay H Mahmoud, and J Mikael Eklund. Blockchain technology in healthcare: a systematic review. In *Healthcare*, volume 7, page 56. MDPI, 2019.

[14] Merlinda Andoni, Valentin Robu, David Flynn, Simone Abram, Dale Geach, David Jenkins, Peter McCallum, and Andrew Peacock. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and sustainable energy reviews*, 100:143–174, 2019.

[15] Wattana Viriyasitavat and Danupol Hoonsopon. Blockchain characteristics and consensus in modern business processes. *Journal of Industrial Information Integration*, 13:32–39, 2019.

[16] Ting Yu, Zhiwei Lin, and Qingliang Tang. Blockchain: The introduction and its application in financial accounting. *Journal of Corporate Accounting & Finance*, 29(4):37–47, 2018.

[17] Le Jiang and Xinglin Zhang. Bcosn: A blockchain-based decentralized online social network. *IEEE Transactions on Computational Social Systems*, 6(6):1454–1466, 2019.

[18] Mayank Raikwar, Subhra Mazumdar, Sushmita Ruj, Sourav Sen Gupta, Anupam Chattopadhyay, and Kwok-Yan Lam. A blockchain framework for insurance processes. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–4. IEEE, 2018.

[19] Ali Alammary, Samah Alhazmi, Marwah Almasri, and Saira Gillani. Blockchain-based applications in education: A systematic review. *Applied Sciences*, 9(12):2400, 2019.

[20] Remya Stephen and Aneena Alex. A review on blockchain security. In *IOP conference series: materials science and engineering*, volume 396, page 012030. IOP Publishing, 2018.

[21] Nicolas Sklavos, Ricardo Chaves, Giorgio Di Natale, and Francesco Regazzoni. Hardware security and trust. *Cham, Switzerland: Springer*, 2017.

[22] Andrew J Paverd and Andrew P Martin. Hardware security for device authentication in the smart grid. In *Smart Grid Security: First International Workshop, SmartGridSec 2012, Berlin, Germany, December 3, 2012, Revised Selected Papers 1*, pages 72–84. Springer, 2013.

[23] Christian Lesjak, Daniel Hein, Michael Hofmann, Martin Maritsch, Andreas Aldrian, Peter Priller, Thomas Ebner, Thomas Ruprechter, and Günther Pregartner. Securing smart maintenance services: Hardware-security and tls for mqtt. In *2015 IEEE 13th international conference on industrial informatics (INDIN)*, pages 1243–1250. IEEE, 2015.

[24] Luc Bouganim and Yanli Guo. *Database Encryption*, pages 307–312. Springer US, Boston, MA, 2011.

[25] Minh-Tuan Truong and Quang-Vinh Dang. Digital signatures using hardware security modules for electronic bills in vietnam: Open problems and research directions. In *Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4.0 Applications: 7th International Conference, FDSE 2020, Quy Nhon, Vietnam, November 25–27, 2020, Proceedings 7*, pages 469–475. Springer, 2020.

[26] Juhyeng Han, Seongmin Kim, Taesoo Kim, and Dongsu Han. Toward scaling hardware security module for emerging cloud services. In *Proceedings of the 4th Workshop on System Software for Trusted Execution*, pages 1–6, 2019.

[27] Ronald Perez, Leendert Van Doorn, and Reiner Sailer. Virtualization and hardware-based security. *IEEE Security & Privacy*, 6(5):24–31, 2008.

[28] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235, 2019.

[29] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, pages 37–42, 2015.

[30] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering*, pages 195–216, 2017.

[31] Saša Baškarada, Vivian Nguyen, and Andy Koronios. Architecting microservices: Practical opportunities and challenges. *Journal of Computer Information Systems*, 2018.

[32] Vladyslav Lysakov, Oleksandr Sievierinov, and Igor Taran. Security of web applications using aws cloud provider. *COMPUTER AND INFORMATION SYSTEMS AND TECHNOLOGIES*, 2021.

[33] Ronald Perez, Reiner Sailer, Leendert van Doorn, et al. vtpm: virtualizing the trusted platform module. In *Proc. 15th Conf. on USENIX Security Symposium*, pages 305–320, 2006.

[34] Maria Fazio, Antonio Celesti, Rajiv Ranjan, Chang Liu, Lydia Chen, and Massimo Villari. Open issues in scheduling microservices in the cloud. *IEEE Cloud Computing*, 3(5):81–88, 2016.

[35] Zheng Li, Maria Kihl, Qinghua Lu, and Jens A Andersson. Performance overhead comparison between hypervisor and container based virtualization. In *2017 IEEE 31st International Conference on advanced information networking and applications (AINA)*, pages 955–962. IEEE, 2017.

[36] Stephen Soltesz, Herbert Pötzl, Marc E Fiuczynski, Andy Bavier, and Larry Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In *Proceedings of the 2Nd ACM SIGOPS/EuroSys european conference on computer systems 2007*, pages 275–287, 2007.

[37] Michael Eder. Hypervisor-vs. container-based virtualization. *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, 1, 2016.

[38] Nuha Alshuqayran, Nour Ali, and Roger Evans. A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 44–51. IEEE, 2016.

[39] Duo Lu, Dijiang Huang, Andrew Walenstein, and Deep Medhi. A secure microservice framework for iot. In *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 9–18. IEEE, 2017.

[40] Long Sun, Yan Li, and Raheel Ahmed Memon. An open iot framework based on microservices architecture. *China Communications*, 14(2):154–162, 2017.

[41] Deeraj Nagothu, Ronghua Xu, Seyed Yahya Nikouei, and Yu Chen. A microservice-enabled architecture for smart surveillance using blockchain technology. In *2018 IEEE international smart cities conference (ISC2)*, pages 1–4. IEEE, 2018.

[42] Simon Heron. Advanced encryption standard (aes). *Network Security*, 2009(12):8–12, 2009.

[43] Evgeny Milanov. The rsa algorithm. *RSA laboratories*, pages 1–11, 2009.

[44] Ronghua Xu, Seyed Yahya Nikouei, Yu Chen, Erik Blasch, and Alexander Aved. Blendmas: A blockchain-enabled decentralized microservices architecture for smart public safety. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 564–571. IEEE, 2019.

[45] Wenquan Jin, Rongxu Xu, Taewan You, Yong-Geun Hong, and Dohyeun Kim. Secure edge computing management based on independent microservices providers for gateway-centric iot networks. *IEEE access*, 8:187975–187990, 2020.

[46] Marc-Oliver Pahl and François-Xavier Aubet. All eyes on you: Distributed multi-dimensional iot microservice anomaly detection. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 72–80. IEEE, 2018.

[47] Marc-Oliver Pahl, François-Xavier Aubet, and Stefan Liebald. Graph-based iot microservice security. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–3. IEEE, 2018.

[48] Stathis Mavrovouniotis and Mick Ganley. Hardware security modules. In *Secure Smart Embedded Devices, Platforms and Applications*, pages 383–405. Springer, 2013.

[49] Chris Mitchell. *Trusted computing*, volume 6. Iet, 2005.

[50] Will Arthur, David Challener, and Kenneth Goldman. *A practical guide to TPM 2.0: Using the new trusted platform module in the new age of security.* Springer Nature, 2015.

[51] Allan Tomlinson. Introduction to the tpm. *Smart Cards, Tokens, Security and Applications*, pages 173–191, 2017.

[52] Liang Gu, Xuhua Ding, Robert Huijie Deng, Bing Xie, and Hong Mei. Remote attestation on program execution. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, pages 11–20, 2008.

[53] Yunlong Guo, Aimin Yu, Xiaoli Gong, Lixin Zhao, Lijun Cai, and Dan Meng. Building trust in container environment. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1–9. IEEE, 2019.

[54] Shohreh Hosseinzadeh, Samuel Laurén, and Ville Leppänen. Security in container-based virtualization through vtpm. In *Proceedings of the 9th International Conference on Utility and Cloud Computing*, pages 214–219, 2016.

[55] Sari Sultan, Imtiaz Ahmad, and Tassos Dimitriou. Container security: Issues, challenges, and the road ahead. *IEEE access*, 7:52976–52996, 2019.

[56] Ramaswamy Chandramouli and Ramaswamy Chandramouli. *Security assurance requirements for linux application container deployments.* US Department of Commerce, National Institute of Standards and Technology . . . , 2017.

[57] Cameron F Kerry and Patrick D Gallagher. Digital signature standard (dss). *FIPS PUB*, pages 186–4, 2013.

[58] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.

[59] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[60] OASIS Standard. Pkcs# 11 cryptographic token interface base specification version 2.40. *Recuperado de https://bit. ly/3EJ18jk [Consultado noviembre 2021]*, 2015.

[61] Tamper resistance {a cautionary note. In Proceedings of the Second USENIX Workshop on Electronic Commerce (1996), author=Anderson, R and Kuhn, M, journal=USENIX Association, pages=1–11.

[62] Sergei Skorobogatov. Physical attacks and tamper resistance. In *Introduction to Hardware Security and Trust*, pages 143–173. Springer, 2011.

[63] Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols: 5th International Workshop Paris, France, April 7–9, 1997 Proceedings 5*, pages 125–136. Springer, 1998.

[64] tpm2 software. Tpm2-software/tpm2-abrmd: Tpm2 access broker amp; resource management daemon implementing the tcg spec.

[65] Will Arthur, David Challener, Kenneth Goldman, Will Arthur, David Challener, and Kenneth Goldman. Tpm software stack. *A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security*, pages 77–96, 2015.

[66] tpm2 software. Tpm2-software/tpm2-tools: The source repository for the trusted platform module (tpm2.0) tools.

[67] tpm2 software. Tpm2-software/tpm2-pytss: Python bindings for tss.

[68] tpm2 software. Tpm2-software/tpm2-tss-engine: Openssl engine for tpm2 devices.

[69] tpm2 software. Tpm2-software/tpm2-openssl: Openssl provider for tpm2 integration.

[70] tpm2 software. Tpm2-software//tpm2-pkcs11: A pkcs#11 interface for tpm2 hardware.

[71] Thorsten Rangnau, Remco v Buijtenen, Frank Fransen, and Fatih Turkmen. Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines. In *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 145–154. IEEE, 2020.

[72] Tuomas Aura. Strategies against replay attacks. In *Proceedings 10th Computer Security Foundations Workshop*, pages 59–68. IEEE, 1997.

[73] Keke Gai, Yulu Wu, Liehuang Zhu, Lei Xu, and Yan Zhang. Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks. *IEEE Internet of Things Journal*, 6(5):7992–8004, 2019.

[74] Shen Wang, Ahmad F Taha, Jianhui Wang, Karla Kvaternik, and Adam Hahn. Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(8):1612–1623, 2019.

[75] Shivam Saxena, Hany EZ Farag, Hjalmar Turesson, and Henry Kim. Blockchain based transactive energy systems for voltage regulation in active distribution networks. *IET Smart Grid*, 3(5):646–656, 2020.

[76] Hossam ElHusseini, Chadi Assi, Bassam Moussa, Ribal Attallah, and Ali Ghrayeb. Blockchain, ai and smart grids: The three musketeers to a decentralized ev charging infrastructure. *IEEE Internet of Things Magazine*, 3(2):24–29, 2020.

[77] EOS IO. Eos. io technical white paper v2. *EOS, Tech. Rep*, 2018.

[78] Moez Krichen, Mariam Lahami, and Qasem Abu Al-Haija. Formal methods for the verification of smart contracts: A review. In *2022 15th International Conference on Security of Information and Networks (SIN)*, pages 01–08. IEEE, 2022.

6

Secure Sensor Prototype Using Hardware Security Modules and Trusted Execution Environments in a Blockchain Application: Wine Logistic Use Case

---

ANTONIO J. CABRERA-GUTIERREZ[1,2], ENCARNACION CASTILLO[2], ANTONIO ESCOBAR-MOLERO[1], DIEGO P. MORALES[2], and LUIS PARRILLA[2].

1. Infineon Technologies AG, Am Campeon 1-15, 85579, Neubiberg, Bavaria, Germany
2. University of Granada, Department of Electronics and Computer Technology, Faculty of Sciences, Granada, 18071, Andalucia, Spain

Journal: MDPI Electronics

- Received 13 June 2023, Revised 4 July 2023, Accepted: 5 July 2023, Published: 7 July 2023.
- DOI: 10.3390/electronics12132987
- Impact factor: Q2
- JCR Rank: 2.9, 131/275 Q2 in the category Engineering, Electrical & Electronic.

Abstract

The security of Industrial Internet of Things (IIoT) systems is a challenge that needs to be addressed immediately, as the increasing use of new communication paradigms and the abundant use of sensors opens up new opportunities to compromise these types of systems. In this sense, technologies such as Trusted Execution Environments (TEEs) and Hardware Security Modules (HSMs) become crucial for adding new layers of security to IIoT systems, especially to edge nodes that incorporate sensors and perform continuous measurements. These technologies, coupled with new communication paradigms such as Blockchain, offer a high reliability, robustness and good interoperability between them. This paper proposes the design of a secure sensor incorporating the above mentioned technologies—HSMs and a TEE—in a hardware device based on a dual-core architecture. Through this combination of technologies, one of the cores collects the data extracted by the sensors and implements the security mechanisms to guarantee the integrity of these data, while the remaining core is responsible for sending these data through the appropriate communication protocol. This proposed approach fits into the Blockchain networks, which act as an Oracle. Finally, to illustrate the application of this concept, a use case applied to wine logistics is described, where this secure sensor is integrated into a Blockchain that collects data from the storage and transport of barrels, and a performance evaluation of the implemented prototype is provided.

6.1 Introduction

In Industrial IoT (IIoT) networks, the extraction of data from the physical to the digital world via sensors is becoming increasingly important [1]. This data collection takes place at nodes distributed within a specific network topology at the edge that are equipped with various sensors that convert physical measurements into digital information. In Industry 4.0, these edge devices are the most important part of the IIoT system because, in addition to extracting data, they can also perform computations at the edge without having to delegate decisions and data flows to a centralised entity [2].

Figure 6.1 shows the typical IIoT architecture, where different layers have been considered. The edge/perception layer is responsible for collecting data from the outside world through different types of sensors. These sensors are usually attached to an IoT device, which has a communication system to send these data or the necessary information to higher layers of the system. The network layer is responsible for sending these data to the various technologies that process the information. Technologies such as Wi-Fi and Bluetooth Low Energy (BLE) are included in this layer, as well as the hardware devices that perform this task: routers, gateways, etc. The processing layer includes the technologies that allow information to be stored and processed, such as databases, data analysis systems or control software. An even higher layer can be defined in terms of system application, where data and information are delivered to the client in different ways depending on the use case [3].

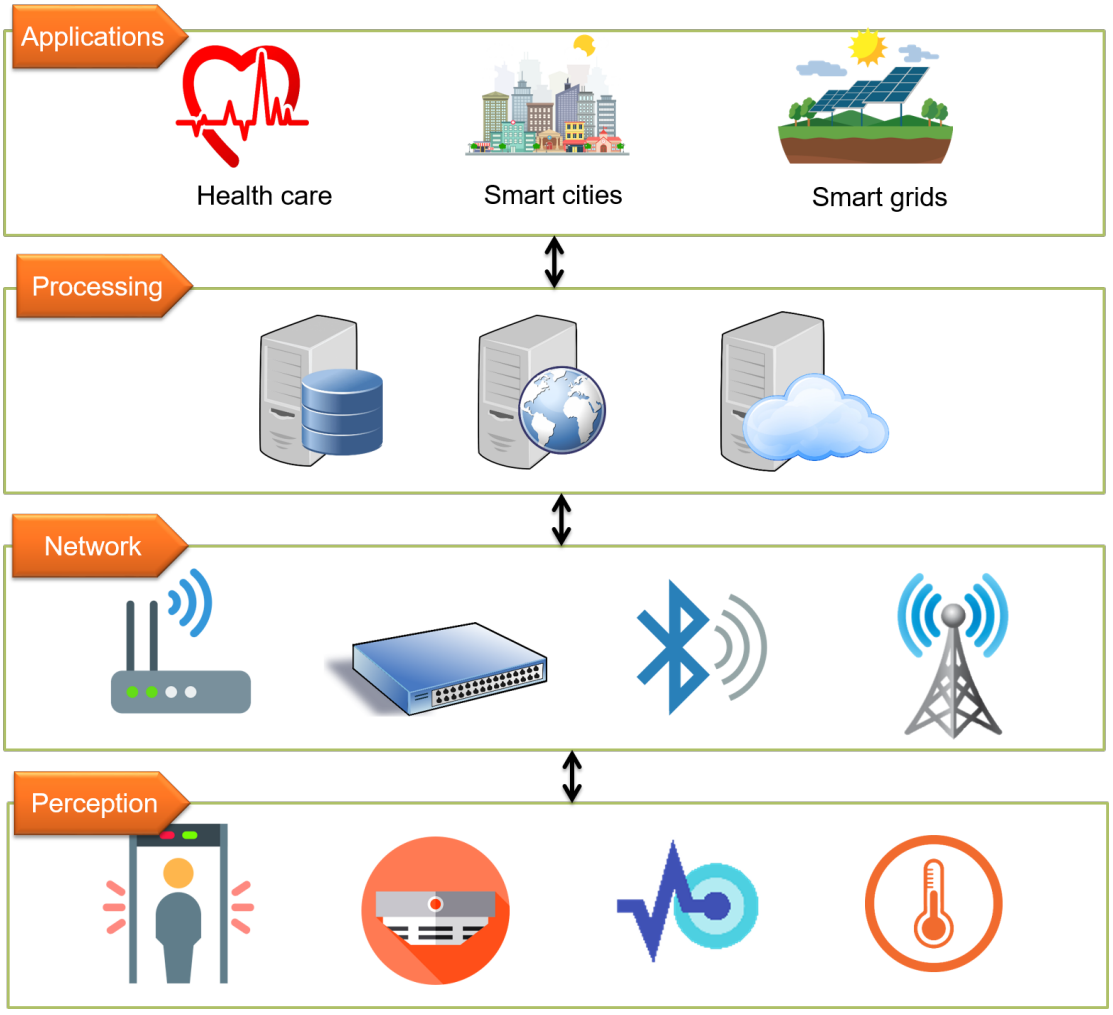

FIGURE 6.1: IIoT architecture.

Nowadays, providing security for IIoT devices and the overall system that they are a part of is critical [4]. At first glance, it might seem that, by providing communications security, the system would be protected, but nothing could be further from the truth. Although one of the main aspects is to protect communications, as these provide the largest attackable area of the system [5], this is not enough. In this type of system, paradigms such as traditional client–server-based protocols and implement access control through a public key infrastructure (PKI) are no longer enough [6].

New Decentralised Ledger Technologies (DLTs) [7]-based paradigms such as Blockchain are becoming a fundamental scheme for ensuring the security of these IIoT systems [8]. Through decentralisation and the underlying cryptography, Blockchain offers secure machine-to-machine communication in which every piece of information or data entered into the network has to be validated by different entities that make up this network [9].

The underlying cryptography in this type of network is normally based on asymmetric cryptography, where Rivest–Shamir–Adleman (RSA) [10] or Elliptic Cryptographic Curve (ECC) [11] algorithms are used intensively. To protect the cryptographic keys, it

is essential that the devices that make up these networks have additional protection at the hardware level.

To protect IIoT devices, technologies such as Trusted Execution Environments (TEEs) [12] and Hardware Security Modules (HSMs) [13] provide firmware and hardware security capable of repelling attacks such as software and physical attacks [14, 15]. HSMs offer rock-solid and certified protection against side-channel attacks, such as fault injection [16]. However, they do not typically support flexible high-level functionality, such as programmable device drivers or secure peripherals. The proposed idea is to combine a non-programmable HSM with a flexible TEE, (i.e., Trusted Firmware-M (TF-M) [17] running in the ARM Cortex-M0 core of a PSoC64 [18]), obtaining the advantages of both solutions.

Designing IIoT devices using TEEs and HSMs is not a trivial task; furthermore, the integration of these devices with Blockchain networks has not yet reached a desired level of maturity [19]. Blockchain networks are increasingly being used and the fields of application are numerous [20–23]. However, there is still much room for improvement in the application of these technologies, as many of them only focus on the deployment and interconnection of devices, creating a decentralised ledger where information can be shared. A typical problem to be addressed is the introduction of data from the off-chain world to the Blockchain world through the so-called Blockchain Oracles [24].

In addition to the design that combines an HSM with a TEE in an IIoT device, this work proposes the integration of the IIoT node within a Blockchain network working as an Oracle. Furthermore, as an application example, the use of this device in a goods logistics use case, such as the transport and storage of wine, is described.

The solution proposed in this article is based on the union of a TEE and an HSM in an IIoT node that fits in Blockchain networks working as an Oracle, providing data securely to the Blockchain. Based on this, this work contributes to create a secure sensor concept in a dual-core platform that extracts data from the physical world (off-chain world) in a secure way. The feasibility of this concept is demonstrated in the application of a use case on wine logistics.

The rest of the article is organised as follows: Section 6.2 describes the different approaches and categorisations regarding security in IIoT nodes. Section 6.3 presents the proposed design of the IIoT node describing the HSM and the TEE used in the prototype and how they work together in order to achieve the desire security level. In Section 6.4, the integration of this IIoT node with the Blockchain is described, with special emphasis on the functioning as a Blockchain Oracle. Section 6.5 presents the use case oriented to the wine logistics, and the entire system is described. Finally, Section 6.6 summarises the conclusion, emphasising the benefits and the applicability of this proposal.

6.2 Related Works

IIoT node design determines the basis on which a system can be built upon. Just as in the construction of buildings, without a good foundation, the building will fall. It is significant that, despite the importance of these devices, there are hardly any specific designs or mentions of the safety of these devices in the literature. In [25], it is concluded that an unsecured hardware platform would lead to an unsecured software stack, highlighting the importance of hardware security. Other studies such as [26] conclude that hardware security is not treated with the same relevance as software security in this kind of system.

Although talking about hardware security is not very generic, there are some studies, such as [27]; in this work, the authors proposed a multi-core intrusion detection for embedded systems. They implemented a secure monitor that continuously monitors the execution behaviour of the controller, which works with an on-chip hardware unit called a timing trace module. In [28], the authors implemented a host-based intrusion detection system in a Field Programmable Gate Array (FPGA) hardware platform that is in charge of detecting attacks in real-time. In the work [29], the authors developed a hardware monitor that operates in parallel to the embedded processor and detects any attack that causes a wrong behaviour. In the recent work [30], the proposed framework (called C4IIoT) helps to detect and mitigate attacks on a complete IIoT system through offloading mechanisms and machine learning techniques. In particular, this work mentions the protection of nodes with HSMs. In [14], an IoT node solution is proposed that incorporates a special type of HSM, a Trusted Platform Module (TPM), for hardware security and integration with the Blockchain.

Apart from C4IIoT, recent studies have emerged, such as: building a complete identity verification framework based on Physical Unclonable Functions (PUFs) [31], FPGA hardware security for data centres [32] and a re-configurable hardware-based isolation and protection mechanism for IoT devices [33].

While there does not seem to be much literature on IoT devices capable of providing device-level security using HSMs, there exists an extensive literature on IoT devices design using TEEs. In this case, works such as [34] propose a solution combining Intel SGX [35] with Bluetooth security. Another solution proposed in [36] offers a system that protects IoT nodes and integrates them to the Blockchain through a TEE. In the work [37], the authors introduced a solution that enables security through ARM TrustZone by encrypting memory and managing access control, protecting sensitive data. The authors of [38] proposed a system for IoT devices that offers security-enhanced attestation for remote terminals based on Intel SGX. In [39], the authors used a TEE (ARM TrustZone [40]) in an IoT device to create a secure enclave. In [41], the authors

introduced a system developed using ARM TrustZone in which the critical applications are running securely and communicate with other parts of the system.

Works such as those cited above highlight several factors: the low cost of the application of TEEs compared to HSMs, the ease of the integration of those in IoT devices since they do not involve introducing hardware changes and, finally, a greater familiarity and acceptance of these technologies by the scientific community, since they are easier to understand, manipulate and modify. However, a promising idea that this work exploits is the combination of these two technologies, which, as will be seen below, offers substantial advantages for creating a robust and secure system. Works combining these two technologies are not very abundant, although some cases can be found.

This is the case of [42], which introduces a design of an IoT node that incorporates ARM TrustZone for attestation; in addition, attested data are secured storage in a TPM. In the proposal [43], a secure logging system from end-to-end between embedded constraint devices and a remote database is implemented, and the architecture design combines a TEE with a TPM.

These proposals that integrate these two technologies are close to the work proposed in this article, but the use of both software and hardware protection is not oriented towards the general purpose of an IoT node but rather focuses on specific mechanisms such as attestation or to protect a logging system. The main innovation of this work on the state of the art described above is the incorporation of a TEE and an HSM in an IoT device to guarantee the reliability and integrity of the data read by the sensors that the device incorporates, creating a root of trust from the source. In addition, integrating these nodes with Blockchain technologies makes the data trustworthy even after they have left the device. To illustrate this, the use case of wine logistics is described.

6.3 Design Proposed

Traditionally, IoT device architectures that incorporate hardware security attach an HSM to the main board that is used to perform the necessary cryptographic operations: encryption of communications, data signing, public–private key generation, etc. In this type of architecture, the microcontroller is in charge of scheduling the necessary cryptographic tasks, delegating them to the HSM and receiving the response from it. On the other hand, the communication is normally carried out through standardised buses such as Inter-Integrated Circuit (I2C) and Serial Peripheral Interface (SPI). In the event that it is necessary to sign the data received by a sensor to ensure their integrity, the microcontroller will receive them from the sensor and then delegate the signing operation to the HSM. This approach, which can be seen in Figure 6.2, shows a serious drawback: when the data arrive to the microcontroller, an attacker can modify the data without

being detected. In the case where the modification is made after the data are signed, this manipulation can be detected at the verification process and the data will be discarded.

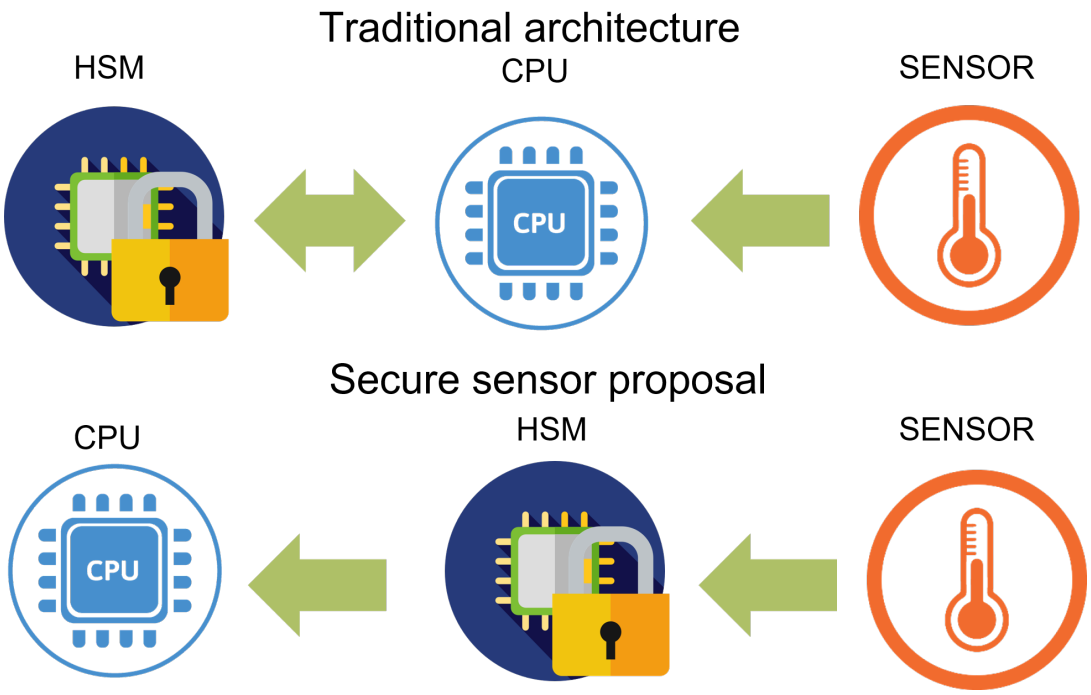

FIGURE 6.2: Design proposed.

In this sense, the secure sensor proposed in this article establishes that the data are sent directly to the HSM and, there, the HSM is in charge of sending it to the microcontroller once it is signed. In this way, if any change is made, it is automatically detected in the verification process, since the data have been previously signed within the secure environment that an HSM offers. Both an HSM and a TEE can play this secure environment role within this proposal.

The design of a hardware-secure IoT node following this concept relies on a careful choice of the platform on which it will be developed. One of the most straightforward choices would be to choose an ARM platform with OS support in order to be able to successfully and easily install a TEE such as the ones discussed above (ARM TrustZone or Intel SGX). However, these types of platforms have some disadvantages that can become a problem in an IoT environment. These disadvantages are related to consumption, since the platforms that support OS, such as Raspberry Pi, usually have a high consumption of energy, and this is a serious drawback when applied to a real environment.

Another requirement to take into account in the design is the ability to modularise the software running on the platform since, on one side, we have to consider the TEE, and on the other side, the rest of the platform firmware. Being able to create isolated execution environments becomes crucial for providing security in this type of architecture where there are two execution environments: a secure processing environment (SPE),

represented by the TEE, and a non-secure processing environment (NSPE), where the rest of the application is executed.

This requirement translates into the use of a platform with more than one core. In this way, a core would be in charge of running the SPE and communicating with the NSPE that is running in the other core.

The platform chosen for this proposal was the PSoC64, which contains two cores: one core is an ARM Cortex-M4 and the other one is an ARM Cortex-M0+. In the Cortex-M4 core, the main application of the device is executed. This has the firmware necessary to carry out the communication with the outside world through a communications protocol that will vary depending on the application (BLE, Wi-Fi, Ethernet, etc.), as well as the libraries and kernel of the embedded OS. The ARM Cortex-M0+ is in charge of running the TEE, which is the TF-M, and incorporates support for the ARM-M microcontrollers family. The communication between the two cores is via an inter-process communication bus. The TF-M isolates the different execution environments (NSPE and SPE) through APIs that are called through the processes that form the system. In addition, the TF-M offers different services such as cryptography and attestation and allows for the execution of secure processes through secure partitions (this will be explained in detail in the following sections). These secure processes allow for the execution of firmware such as sensor drivers or communication protocols. Figure 6.3 illustrates this design.

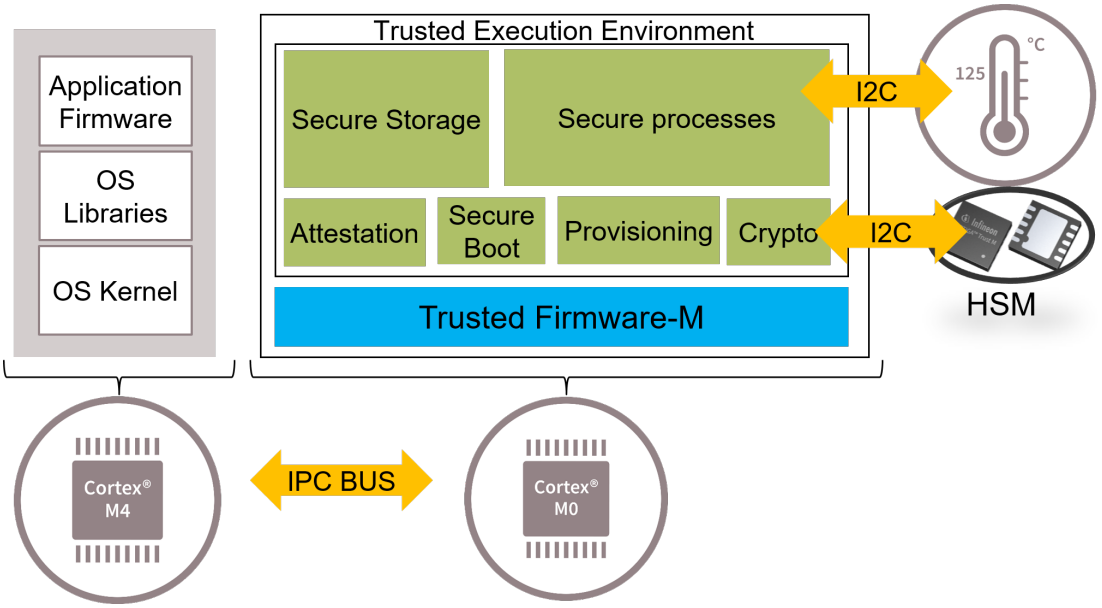

FIGURE 6.3: Secure sensor concept.

In this way, the driver of the HSM is implemented within the TEE, improving the reliability of the accesses to the HSM functionality. The functionality of the HSM can only be used from the NSPE (ARM Cortex-M4) through a controlled and predefined API by the TF-M. Additional mechanisms can be integrated in the TEE, such as a sensor driver through a secured I2C peripheral, enabling high-level and high-value functionality,

such as an API request to obtain the value of the sensor directly signed by the HSM from the NSPE, greatly limiting the attack vectors and tampering with possibilities of the sensor value from the NSPE.

The design of this architecture was implemented in the CY8CKIT-064S0S2-4343W evaluation board; specifically, the MCU used was the CYS0644ABZI-S2D44, which has two cores: an ARM Cortex M4 working at 150 MHz and an ARM Cortex M0+ at 100 MHz. Regarding the sensor, in this proposal, we attached an S2GO pressure DPS310 board by Infineon [44] to the PSoC64. This board is equipped with one DPS310 barometric pressure sensor, offering a high pressure resolution (±0.005 hPa) and temperature accuracy of approximately ±0.5 °C. For communication, the board provides an I2C interface that connects the DPS310 sensor with PSoC64. The HSM used was the Infineon OPTIGA Trust M [45], which has the functionalities for private and public key generation, signing and encryption. It also has different power saving modes, which makes it perfect for this kind of IoT application.

6.3.1 Trusted Firmware-M

TF-M is the TEE implemented in the Cortex-M0+ of the PSoC64. The implementation is aligned with Platform Security Architecture (PSA) certified guidelines and offers isolation between NSPE and SPE. TF-M consists of several modules: secure boot to authenticate the integrity of NSPE and SPE images, isolation between environments, communication between SPE and NSPE and modules that enable cryptography, internal secure storage, attestation and secure services.

Figure 6.4 shows how these modules are distributed as well as the isolation level that TF-M provides. The first level of isolation, represented in Figure 6.4 with a black solid line, separates the NSPE from the SPE. This limit means that applications running on the NSPE do not have access to the secure core where the TF-M runs, thus only through a defined API (PSA API) can these applications make calls to the services offered by the TF-M. The applications provided by the TF-M are divided into two types: PSA Root of Trust (RoT) and applications RoT. Between these two types, there is another level of isolation established by the TF-M. This level 2 isolates the services running in the application RoTs from those running in the PSA RoT, making these services run independently of each other, unless communications are established between them through certain APIs (service API and PSA secure partition API). The reason for why this division exists is that the services provided by PSA RoT are already pre-established by the TF-M and the services provided by the application RoT are programmable and modified by the programmer through the creation of secure partitions. The services predefined by the RoT PSAs are as follows:

- Cryptography: This service implements cryptographic operations, including symmetric ciphers, asymmetric algorithms, hash algorithms, key derivation and random number generation.
- Attestation: This service reports the immutable device identity and boot state.
- Internal trusted storage: This service provides secure storage of small data items such as cryptographic keys.
- Firmware update: This provides firmware image update functionality by retrieving image information about the firmware images stored on the device memory. This service validates the image and trigger a reboot to restart the platform.

This last PSA RoT service is strongly related to the secure boot module that TF-M implements. This module is essential for security to enforce firmware authenticity to protect it against malware execution. This is achieved by building a chain of trust where each step in the execution chain authenticates the next step. The chain of trust is based on an RoT that is implemented using asymmetric cryptography. The RoT is a combination of an immutable bootloader and a public key. The bootloader is started when the CPU is restarted, running in secure mode and authenticating the firmware image by means of hash validation and an RSA signature. The public key, with which the checks are performed, can be embedded into the bootloader image or can be provided to the chip during manufacturing. In case of successful authentication, the bootloader passes the execution to the secure image.

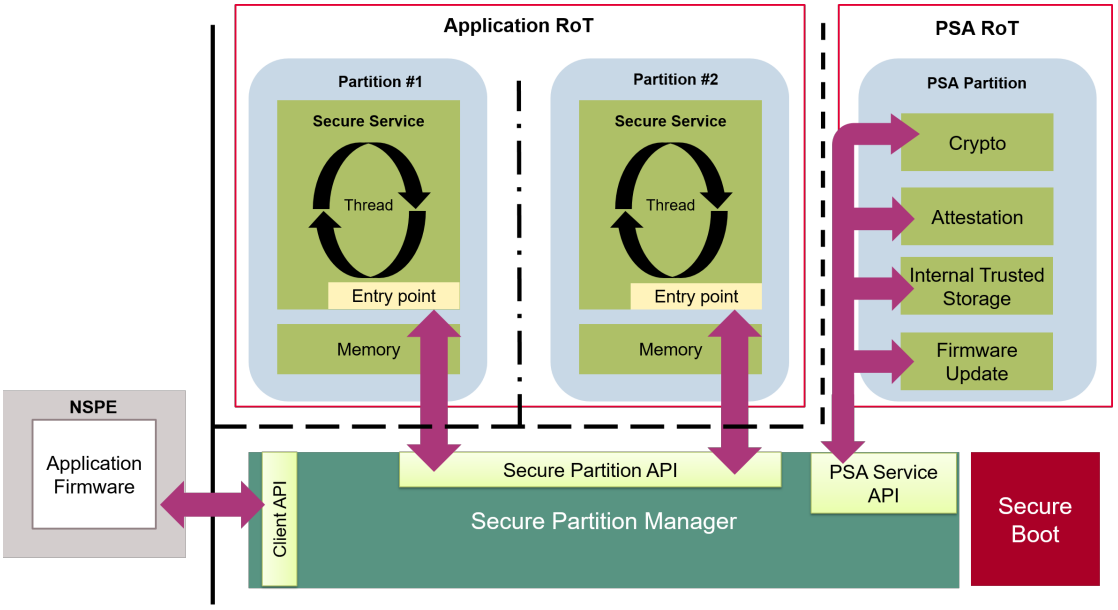

FIGURE 6.4: Trusted Firmware-M.

RoT applications are developer-defined applications that are installed on secure partitions. Each secure partition is a single thread of execution and is the smallest unit

of isolation. If the isolation level 3 is implemented, every secure partition is isolated from each other. Each secure partition can host one or more application RoT service, which typically shares functionality or data. Secure partitions have a persistent identifier, called a partition ID, that can be used for access control within the system. These identifiers must be unique within the device, protected by the Secure Partition Manager (SPM) at runtime, and unchanged when firmware is rebooted or updated.

The SPM is the most privileged firmware, providing the fundamental security services to secure the PSA root of trust and enabling isolated firmware components to communicate between them through APIs (client API, secure partition API and PSA RoT services API). In addition, SPM implements the scheduling logic to ensure that the request is delivered and processed correctly. Finally, one of the most important tasks of the SPM is to access secure peripherals and handle interrupts.

Secure peripherals are critical in this proposal since two hardware devices (atmospheric pressure sensor DPS310 and OPTIGA Trust M) are being used through secure services implemented in the TF-M. These devices are connected to the Cortex M0+ using an I2C interface and while the data are served to this core, the TF-M must implement certain mechanisms to make the communication bus blocked or inaccessible to the other core. In addition, each peripheral has an associated driver and one or more entries in the device's interrupt table. These interrupts must be managed by the TF-M in order to avoid other cores managing them.

For addressing these issues and creating a secure peripheral, the TF-M creates secure Memory Mapped I/O regions (MMIOs) that exclude access to these regions for other secure or non-secure processes. Thus, when a secure process occupies the peripheral, it excludes any other process, secure or not, from accessing it at the same time.

Thus, the implementation of these secure services is based on a driver in which interrupts are handled through the SPM and a memory area (MMIO) from which the memory registers used by the driver are mapped. This driver is implemented in a thread that is in an infinite loop waiting for a call through an entry point that communicates through the secure partition API. The partition name, the interrupts, the MMIO and the secure services on which a partition depends are registered in a partition manifest to which the SPM associates a service ID (SID). The SPM associates an SID to the secure service implementing the HSM driver as well as to the secure service implementing the pressure sensor driver. In Figure 6.5, the flowchart is shown from when the non-secure core requests data to the core where the TF-M is running. This diagram shows how the program jumps between different levels of isolation by means of the diverse APIs available, until it executes the driver peripheral through the entry point. This peripheral can be the HSM or a sensor.

In this way and through these mechanisms, the drivers of the two hardware devices connected to the Cortex M0+ are implemented in a secure environment isolated from other

processes that may interfere with the data flow and the I2C communication protocol, creating a secure peripheral in which the data received are reliable.

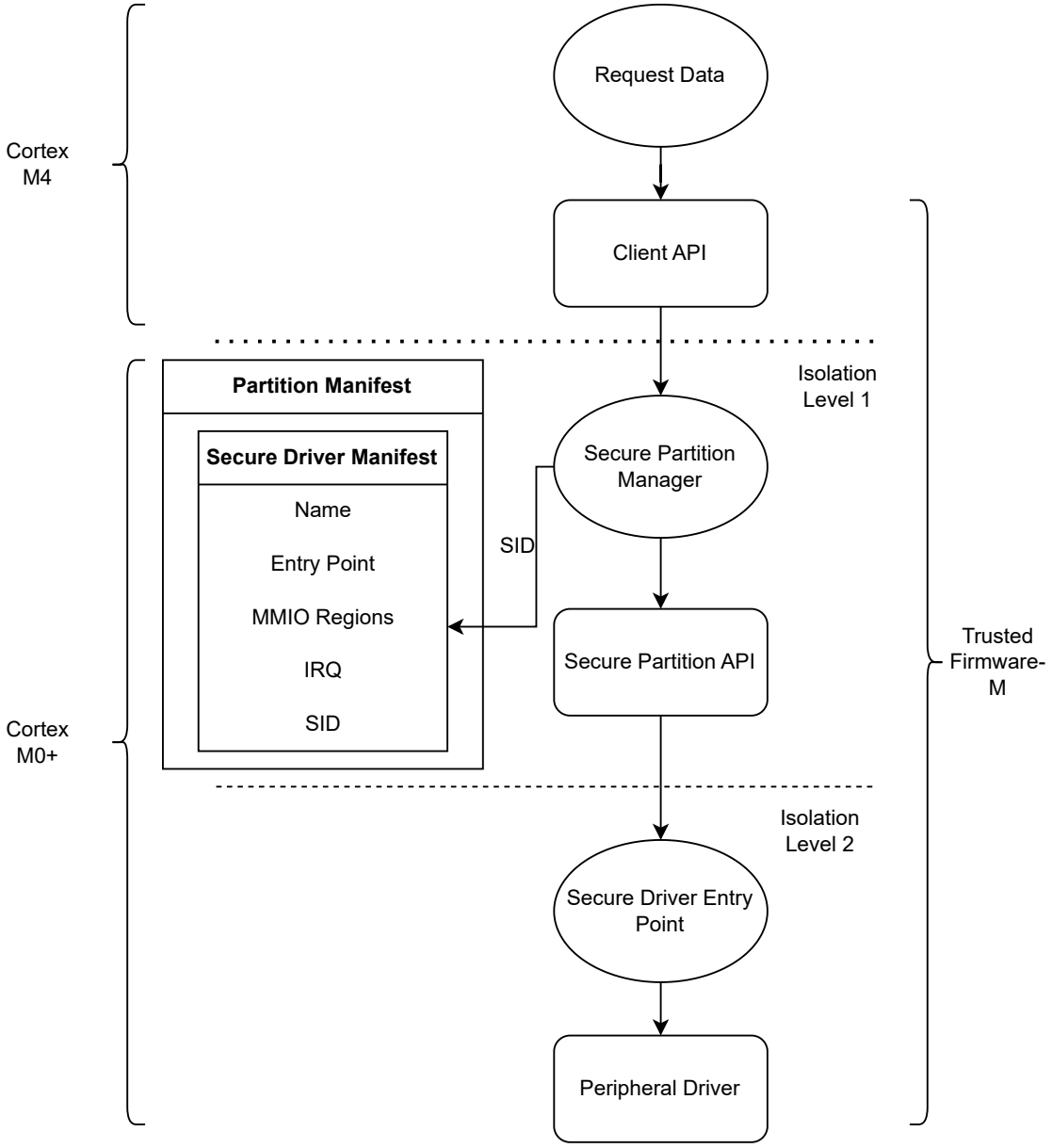

FIGURE 6.5: Driver implementation in TF-M.

6.3.2 Hardware Security Module

In order to maintain the integrity of the data once they leave the secure core, they must be signed. Although signing can be performed through the PSA RoT cryptographic service, this entails certain risks as discussed above. Thus, including an HSM in this way offers extra protection to the system by adding a layer of hardware security. OPTIGA Trust M performs the functions of a cryptographic service implemented in a secure

partition, where it performs key generation, public key export and hash and signing operations.

Figure 6.6 shows the order of operations within the execution flow of the secure core processes. Key generation is performed through a True Random Number Generator (TRNG) that incorporates the HSM, which generates true random numbers from high-entropy microscopic physical events. The signature mechanism is based on an Elliptic Curve Digital Signature Algorithm (ECDSA) [46], which uses ECC keys for the signature; in particular, in this proposal, the curve is the secp256r1, also known as NIST P-256. Key generation is carried out in the startup and the private key is stored permanently in the HSM. During the commissioning process, the public key is exported from the secure sensor to the entity, which verifies the data signatures made during the operational mode of the device.

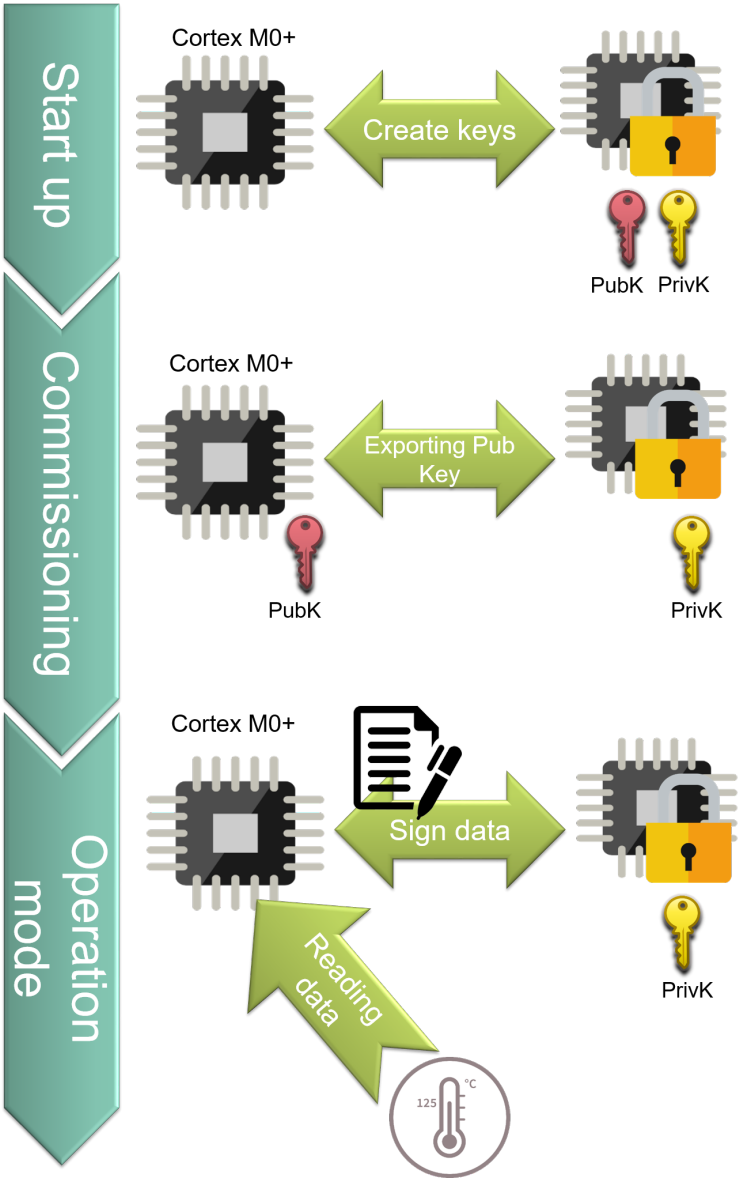

FIGURE 6.6: Operations carried out by the HSM.

In this way, through the generation and secure storage of the private key within the HSM, the secure sensor cannot be impersonated by extracting the private key or duplicating it.

This secure sensor concept fits into various IoT architectures and can be integrated with technologies used in this field, but there is one type of technology where this proposal has a fist-in-glove fit. This is the case of Blockchain technologies.

6.4 Blockchain Application

The application of the secure sensor concept in an IoT node offers, as mentioned above, several advantages. In any system where it is applied, the secure sensor is able to provide secure data by signing and verifying the data, ensuring data integrity.

Systems where the secure sensor is implemented must have a PKI system where an IoT node implementing the secure sensor approach must be enrolled. In this case, the node that provides data to the system and therefore signs it must be registered in the same PKI as the node that receives those data and therefore verifies them. This process of signature and verification must take place in all environments and systems where this concept is applied. However, as stated in Section **??**, PKI paradigms are no longer enough in IIoT environments [6]. New paradigms such as Blockchain have emerged, providing further security benefits to IIoT networks [**? ?**]. In this situation, the integration of this type of technology with the concept of a secure sensor becomes fundamental [**?**]. In this case, it depends on the architecture and topology of the Blockchain network to incorporate a PKI-based access control system. In any case, Blockchain networks have mechanisms to decentralise PKI through different entities and based on smart contracts [47].

Traditionally, private and public keys have been used to enrol the users and/or devices into a network by using a PKI combined with a symmetric cipher. In the case of systems implementing Blockchain, these keys are also used for signing and verifying transactions and, additionally, to register and to provide the permissions required prior to authentication in private Blockchain networks. Due to the involving of all these operations, the protection of cryptographic keys in an HSM becomes especially relevant in Blockchain networks since the extraction of these keys can lead to the impersonation of certain nodes and the fraudulent sending of transactions that, if they are subsequently verified, are impossible to undo.

In addition to the integration of the secure sensor in a PKI implemented in Blockchain, this concept can also provide reliable data to the Blockchain functioning as an Oracle.

6.4.1 Working as an Oracle

In Blockchain terminology, an Oracle is the entity that provides reliable data to the Blockchain network. These data are by definition reliable and network participants are obliged to use them without doubting their veracity as they have been provided by the Oracle.

This component of Blockchain networks, although certainly neglected in the literature [48, 49], is of crucial importance for the reliability of this type of network. The vast majority of Blockchain networks use data coming from the off-chain world, and the untruthfulness of these data can cause important failures in these systems as well as erroneous ledger states.

There are different types of Oracles [50] that, depending of the information source, can be classified into three groups: software Oracles, where data come from online sources, hardware Oracles, which obtain the data from the physical world—for example, from sensors—and human Oracles, for which the data come through a human, who enters and verifies information, e.g., financial data. A classification can also be established depending on the trust model [51]:

- Centralised trust model: This Oracle model is based on an entity that certifies the reliability of provided data.
- Decentralised trust model: Decentralised Oracles avoid the single point of failure but they add an extra overhead since, being based on several entities, the latency increases as a consensus protocol between them is required.

According to these established classifications, in the proposal of this work, the Oracle would correspond to a hardware type, as it would collect data from sensors, and a centralised trust model, as it is a single entity that provides the data to the Blockchain.

The signature of the data is performed at the extraction of the data in the secure core so that the data can hardly be modified before the signature. The keys are stored in the HSM, which offers tamper-resistant protection, and the communication with the sensor is carried out through the secure core. Once the data have been signed, and therefore any modification to them is detected in a subsequent verification, they leave the secure core to be collected in the NSPE. The NSPE performs the communication functions with the Blockchain, delivering the data to the entity that has requested them, which performs this verification using the public key before introducing the data into the Blockchain through a transaction. The public key is available through a certificate that has every entity of the network and has been issued by a Certification Authority (CA) that belongs to the Blockchain or is implemented through the Blockchain.

In this way, the Oracle that implements the secure sensor proposal is able to deliver data to the Blockchain, ensuring that any modification of the data is detected. While guaranteeing the integrity and authenticity of the data, there are possible attacks that must be considered in this concept, such as replay attacks, and how they can be mitigated using timestamps.

6.4.1.1 Timestamping

To prevent an attacker from obtaining the data and sending them as the data required by the receiver at that precise moment, a timestamping mechanism must be implemented.

Before the signed data leave the secure environment, a timestamp attached to the data is needed so that once the entire packet (data plus timestamp) is signed, the receiver can perform a verification that the timestamp corresponds to the time at which the request for the data was made.

The logical solution would be to synchronise the clocks of both the transmitter and the receiver. This solution, although the simplest, has different problems, which are that modifying a clock in a hardware device is very simple and that it is a very typical attack in this type of device [52].

The solution proposed in this work relies on obtaining the timestamp from a reliable external source; thus, two proposals are given below:

- Using Network Time Protocol (NTP) [53]: An NTP provides the basic protocol mechanisms necessary to synchronise the time of different systems to an accuracy of one nanosecond. It is easily scalable and, through handshaking between the NTP server and the client, the clocks are synchronised. In order to enhance the security of this protocol, it is proposed to use the Network Time Security (NTS) protocol, which enables NTP clients to be cryptographically identified against NTP servers to ensure the authenticity and integrity of exchanged time packets [54]. This solution, although efficient and independent of the secure sensor operation process with the Blockchain, requires an NTS server in the system where it is implemented.
- Using the timestamp from Blockchain's block: Blockchain networks, when committing a new block to the chain, include a timestamp in the block. This timestamp is a reliable source in order to be able to synchronise the two communicating devices. However, this timestamp is discrete, i.e., it does not count the time sequentially as a clock would, but seconds may have passed between the consulted block and the next one, as in the case of Ethereum [55], or even minutes, as in the case of Bitcoin [**?**]. To solve this, a mechanism is presented as shown in Figure 6.7. When the Blockchain client requests data from the secure sensor, it queries the timestamp of the last Blockchain block (T'). Before sending the signed data, the secure

sensor asks for the timestamp of the last block of the Blockchain (T). When the data and the signed timestamp reach the client, the client queries the timestamp of the last block (T''). To decide whether these data are accepted or discarded, the following equation has to be fulfilled:

$$T' \leq T \leq T'' - \varepsilon \tag{6.1}$$

ε is a variable that sets the acceptance tolerance of the timestamp and is related to the average generation time of a block on the Blockchain; therefore, it depends on the Blockchain where the secure sensor is used and it will never be equal to or greater than this averaging time. This solution does not require any external infrastructure to obtain the timestamps as they come from the Blockchain.

This method, although obtaining timestamps from a more reliable source than the previous proposal, works better with a high average number of generated blocks than with a low one. It can be concluded that the selection of one of these two solutions is determined by the type of Blockchain that is applied: in the case of a Blockchain such as Ethereum, where the number of blocks per minute is high, the second method is more convenient, while in the case of a Blockchain such as Bitcoin, where the number of blocks per minute is low, using the second method would add nothing to the reliability of the data as even the three timestamps can be the same within a 10 min interval, for example; therefore, the first method would be more appropriate.

Another alternative solution is to adopt a hybrid one that captures both the benefits of using the first approach (more granular timestamps) and the second (more reliable timestamps).

It should be noted that the processing and delay times within the network would be negligible since it is assumed that these times are constant and that there is no large abrupt variation during the timestamp acquisition mechanism as exemplified in Figure 6.7. In the case of a sharp increase in these times, the timestamp (T) would no longer be valid in the case where it does not fulfil Equation (1). In such a scenario, ε would mark the threshold of acceptance or rejection. This variable will be set depending on the environment where this mechanism is implemented. Applying the hybrid approach mentioned before, it is possible to solve this kind of problem by performing different timestamp verification: through the last block of the Blockchain and through an NTP server.

Finally, it is worth mentioning that these processes must be implemented in the secure core since an implementation in the NSPE can lead to malicious modifications of the timestamp by an attacker before signing.

Having described the design of the secure sensor and its integration in Blockchain networks working as an Oracle, it remains to show its applicability in a real and disruptive use case.

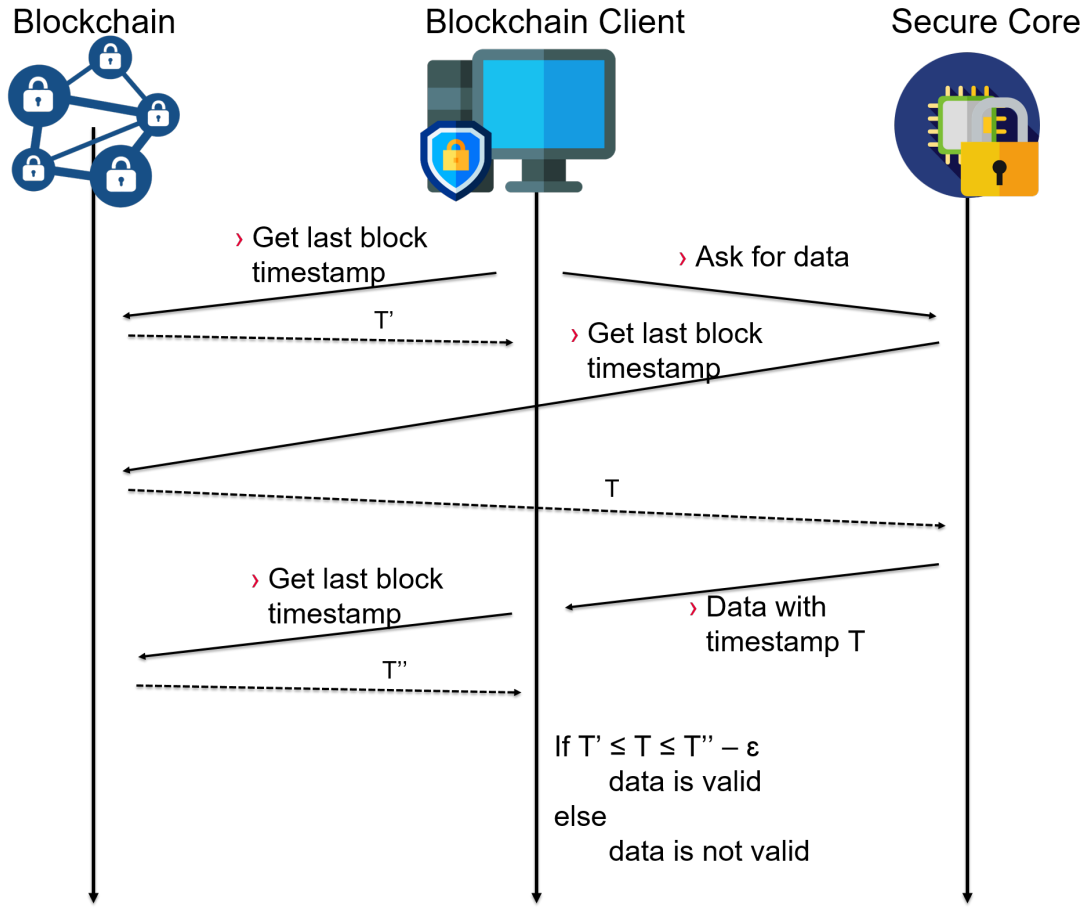

FIGURE 6.7: Timestamp mechanism.

6.5 Wine Logistic Use Case

The quality of a wine can be noticed by its taste and flavour. From the cheapest to the most expensive wines, a failure in the production, transport or sales chain can ruin the product. Traditionally, wine is stored in a cellar with temperatures between 10 and 16 °C [56]. Exposure to temperature or humidity spikes can cause spoilage such as fungus or maturation interruption [57]. Despite the importance of the temperature and humidity for the quality of the wine, this information is not generally shared with the customer.

The use of Blockchain in this type of application, as well as the secure extraction of data through sensors, makes the concept proposed in this article fit perfectly in this use case. Blockchain works very well in logistics use cases [58], where data are regularly collected to create a record. Thanks to these data stored in a decentralised database, the records are immutable and, through smart contracts, it is possible to automate processes such as the devaluation of the price of wine if it exceeds a certain temperature for a certain period time. To achieve this, an architecture was built based on the secure sensor and Blockchain, incorporating several HSMs.

6.5.1 System Description

The system is compounded by an edge node sensor platform based on a PSoC64, as is shown in Figure 6.8. Using the secure sensor concept with its secure booting assures the integrity of the firmware running on both cores. One core is used as a security co-processor, where an HSM is used to store the key material, providing additional protection against physical attacks. The drivers of the HSM and the sensor are programmed in the secure core, which is in charge of both gathering and signing the sensor value, assuring its integrity after it leaves the secured environment. The value is then sent to the Amazon Web Service (AWS) cloud, where a Blockchain based on Hyperledger Fabric (HLF) [59] guarantees a trusted traceability: abnormal events can trigger alerts in smart contracts, automating business logic and dramatically reducing operating costs, delays and the possibility of fraud. After the wine is bottled, the customer can read the secure and unique ID of each bottle via an HSM-based Narrow Field Communication (NFC) sticker, which is then used to retrieve all its historical data from the Blockchain. Figure 6.9 shows the whole architecture.

The node consists of the DPS310 atmospheric pressure sensor, the HSM OPTIGA Trust M and an LTE communication shield providing an antenna. In this way, the data are sent through this channel to the cloud, where an instance of HLF runs. The node incorporates a shield to attach sensors such as the DPS310. In Figure 6.8, only two attachments are shown (the DPS310 and the HSM), as well as a free socket where one can attach the humidity sensor, a GPS or any other type of sensor.

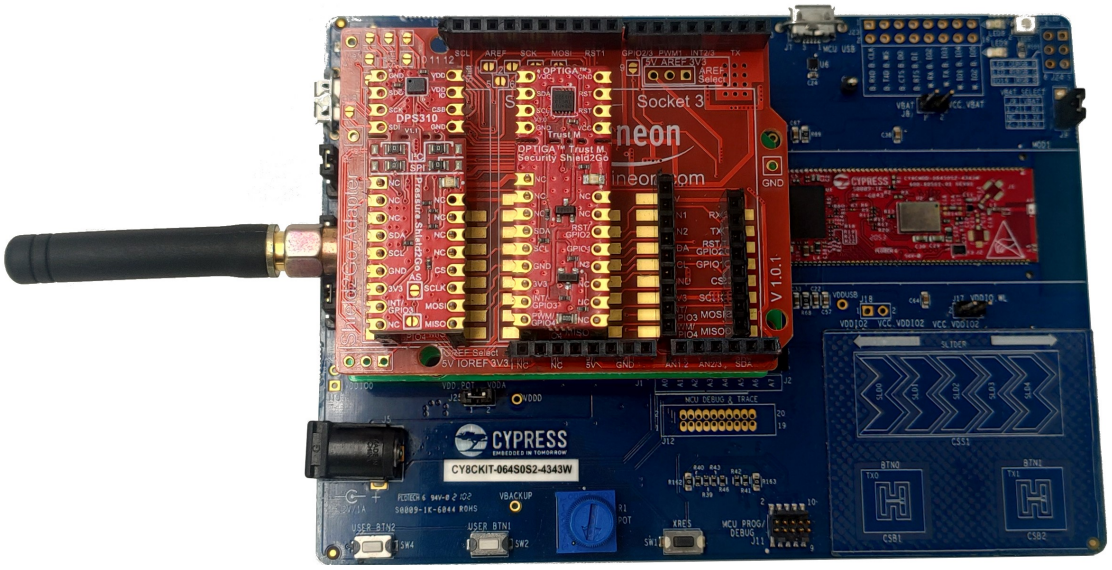

FIGURE 6.8: PSoC64-based edge node.

This node is located in the barrels to monitor the temperature and humidity parameters during transport and storage as shown in Figure 6.10. Each edge node located in a barrel contains a digital ID stored in the HSM. This identifier is associated with the ID

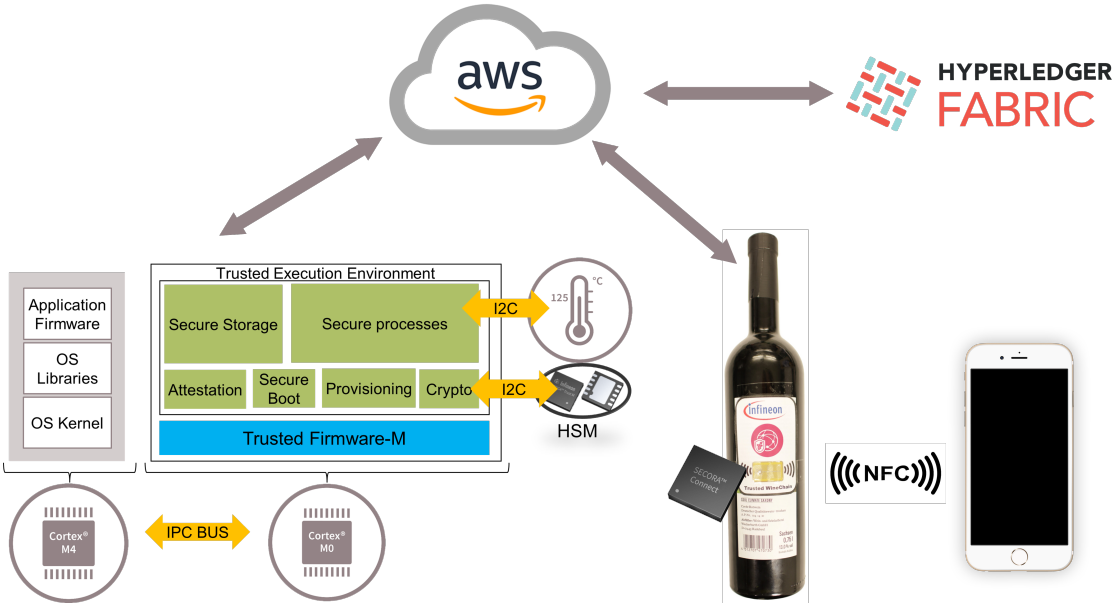

FIGURE 6.9: Architecture proposed in wine logistic use case.

contained in the HSM-based NFC sticker of the bottles that are filled with wine from the barrel. By associating these IDs, it is possible to keep track of the information associated with wine bottles from purchase in the supermarket to direct bottling in barrels. As well as storing sensor data related to wine on the Blockchain, the different entities that make up this network can also store additional information related to wine production: type of grape variety, type of harvest, parcel number, etc.

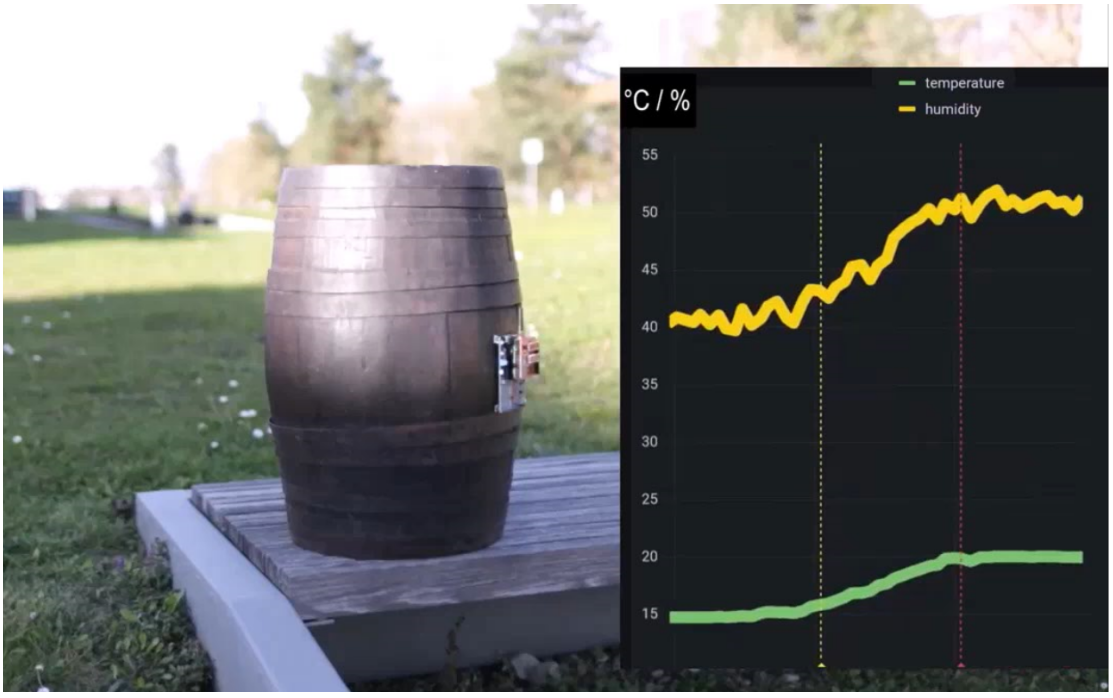

FIGURE 6.10: PSoC64 located in a barrel with the data records gathering by sensors.

When wine is stored in barrels, each barrel is assigned a unique digital ID that is generated through a private key stored in the HSM. During transport, the node takes periodic measurements of the environmental conditions through the node's sensors. These data are collected and stored in the ledger. When the barrel is uncorked and its contents are poured into different bottles, a cryptographic key derived from the digital ID of the barrel is imported into each bottle's HSM. Thus, the keys in the bottle are associated to one cryptographic key in the barrel as well as the data registry recorded by the sensors. Table 6.1 shows how the data structure is stored in the ledger.

TABLE 6.1: Blockchain ledger example.

BarrelID	Derived Key1	Derived Key2	Derived KeyN	Data[]: {Timestamp, Position, Temperature, Humidity}
ZHm...hZnZG	RmYXz...hc2Rm	MjMxN...3NmZm	ODc5...hkYWN	[{1687886326, 31.95162, −28.12310, 28.4, 12%}, ...]
cnR...mRzZG	2Fmc2...0ZXJ3	M2Vyd...RjdGy	MjNb...HV6dc	[{1687886010, 1.95165, −10.23433, 10.1, 29%},...]
ODc...NjU0y	aW9rm...a8O2a	NDMgY...Z2hc2	NDMy...zZnJi	[{1687800125, 7.35234, 20.45643, 35.9, 45%},...]

The digital IDs of the barrels are 32-byte base64-encoded hashes, as well as the derived keys for the bottles. The sensor data are associated to the barrel digital ID, thus forming an array where in each position there is a measurement identified by a timestamp. The components of the Blockchain deployed depend on the number of organisations involved in the business model. If it considers a transport company plus different storage companies (supermarkets, warehouses, etc.), each of them will have a copy of the ledger that will be updated over time. Each update will be periodically triggered through a Blockchain transaction and validated by the different peer nodes of each organisation.

In this way, thanks to the Blockchain and embedded HSMs in the edge nodes and in the stickers' bottle, information is stored securely, providing a robust and reliable system, automating processes through smart contracts and delivering information more clearly and accurately to customers who, through a simple smart phone with built-in NFC, can access this information. The secure sensor acting as an Oracle sends the signed information to the Blockchain, where it is further validated before entering the ledger. Using this architecture ensures that the data entering the ledger are reliable from the moment they are extracted by the sensors, and that they have not been modified either during processing at the edge node or when they are stored in the ledger, as their veracity at that moment is approved by the ledger members.

6.5.2 Performance Evaluation

This section contains the experimental performance measurements carried out for the concept of secure sensor described above. In this sense, the performance evaluation was focused on the processing time required by the implementation of a secure sensor on the PSoC64, from the moment when the sensor data are read until they are sent to the non-secure core for subsequent transmission. For this reason, the tests performed on the communication system and the Blockchain were discarded, as they depend on the

communication protocol used as well as the nodes deployed in the Blockchain network for AWS.

Measurements were carried out on the PSoC64 platform, specifically using the MCU CYS0644ABZI-S2D44, which has two cores: an ARM Cortex M4 working at 150 MHz and an ARM Cortex M0+ operating at 100 MHz. The TF-M is running in the M0+ and an HSM (OPTIGA Trust M) is attached to this core. The sensor used in this evaluation was the Infineon DPS310 atmospheric and temperature sensor. The measurements of this secure sensor concept were compared to the traditional concept shown in Figure 6.2. In this concept, the MCU requests the data from the sensor, and when they are returned to the main MCU, they are subsequently signed by the MCU using an HSM. This approach has a variant in which a software-simulated HSM is used, i.e., the signature and key creation operations are carried out using C libraries implemented in the PSoC. Both of these approaches are much less secure as the data arrive unsigned at the MCU and no use is made of the TEE to secure the peripherals. The times shown in Table 6.2 are the average of a sample of 20 measurements. Times are in milliseconds.

TABLE 6.2: Performance evaluation in milliseconds comparing three approaches.

	Traditional Approach with Software HSM	**Traditional Approach with Hardware HSM**	**Secure Sensor Approach**
Key Generation	9450	2900	2945
Signature Algorithm	5237	310	350
Sensor Read	43	43	85
Total Time	15.305	3810	4160
Total Time without Key Generation	5387	423	518

From Table 6.2, it is shown that the fastest option is the one using hardware HSM in the traditional architecture. This is mainly because the use of a TEE such as the TF-M is a communication time overhead as it incorporates more complex software mechanisms. Figure 6.4 shows all the modules and how they communicate with each other. This is a significant increase in communications and the TF-M protects the sensor driver and the HSM, creating a secure peripheral.The advantage of performing operations using an HSM is the speed compared to the software approach. In short, the secure sensor approach is 1.09 times slower than the traditional approach using HSM, but is 3.67 times faster than the traditional approach using a software HSM. Although the secure sensor approach is not the fastest, it is the most secure approach as the TEE implemented in the PSoC64 introduces security mechanisms such as the secure peripherals discussed earlier in the manuscript.

6.6 Conclusions

This paper proposes a secure sensor design and implementation using different mechanisms for providing a data root of trust. By combining a TEE and an HSM in a dual-core microcontroller with a secure sensor driver implemented in the secure core, the immutability and integrity of data are achieved. The main goal of the proposal is the use of the secure sensor concept, where a secure core is located between the main microcontroller and the sensor. In the secure core, both the HSM and TEE work together, obtaining the advantages of both solutions. HSMs offer hardware and certified protection against physical attacks. However, they do not typically support flexible high-level functionality, such as programmable device drivers or secure peripherals, features provided in this solution by the TEE. Combining an HSM with a flexible TEE, running in the secure processor, a secure environment is created where sensor drivers can be implemented, protecting the communication bus from software attacks as well as providing the ability to sign the data received by the HSM before the data leave the secure core.

The design of this secure sensor fits the role of a Blockchain Oracle, capable of providing reliable data to the ledger thanks to the incorporation of timestamping mechanisms to provide reliability against replay attacks. This fills a gap in the literature regarding Blockchain Oracles, which are rarely addressed.

The feasibility of this approach in a real use case is demonstrated in a wine logistics scenario, in which the secure sensor acts as an Oracle sending temperature and humidity data to the Blockchain. Thanks to these records stored in the Blockchain, customers can obtain accurate information about these products and, through smart contracts, automate business logic and reduce operating costs, delays and the possibility of fraud.

The incorporation of this secure sensor proposal into Blockchain technologies as an Oracle role makes this concept a promising proposition for the future, adding security and integrity to the data required by scenarios as a wine logistic chain. Whether in Blockchain or non-Blockchain networks, the secure sensor solution fills a gap in the security of systems that make intensive use of data from the outside world.

6.7 Acknowledgement

This work was supported by Infineon Technologies AG. Additional funding was provided by European Union's Horizon Europe research and innovation program through the funding project "Cognitive edge-cloud with serverless computing" (EDGELESS) under grant agreement number 101092950. The work has also been funded by FEDER/Junta

de Andalucia-Consejeria de Transformacion Economica, Industria, Conocimiento y Universidades under Project B-TIC-588-UGR20.

6.8 References

[1] Payam Barnaghi, Martin Bauer, Abdur Rahim Biswas, Maarten Botterman, Bin Cheng, Flavio Cirillo, Markus Dillinger, Hans Graux, Seyed Amir Hoseinitabatabaie, Ernö Kovacs, et al. Iot analytics: Collect, process, analyze, and present massive amounts of operational data–research and innovation challenges. In *Building the Hyperconnected Society-Internet of Things Research and Innovation Value Chains, Ecosystems and Markets*, pages 221–260. River Publishers, 2022.

[2] Soumyalatha Naveen and Manjunath R Kounte. Key technologies and challenges in iot edge computing. In *2019 Third international conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC)*, pages 61–65. IEEE, 2019.

[3] Mohammad Ali Jabraeil Jamali, Bahareh Bahrami, Arash Heidari, Parisa Allahverdizadeh, Farhad Norouzi, Mohammad Ali Jabraeil Jamali, Bahareh Bahrami, Arash Heidari, Parisa Allahverdizadeh, and Farhad Norouzi. Iot architecture. *Towards the Internet of Things: Architectures, Security, and Applications*, pages 9–31, 2020.

[4] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar. A survey on iot security: application areas, security threats, and solution architectures. *IEEE Access*, 7:82721–82743, 2019.

[5] Hichem Mrabet, Sana Belguith, Adeeb Alhomoud, and Abderrazak Jemai. A survey of iot security based on a layered architecture of sensing and data analysis. *Sensors*, 20(13):3625, 2020.

[6] Sandeep Mathur and Ankita Arora. Internet of things (iot) and pki-based security architecture. In *Industrial internet of things and cyber-physical systems: transforming the conventional to digital*, pages 25–46. IGI Global, 2020.

[7] Claudia Antal, Tudor Cioara, Ionut Anghel, Marcel Antal, and Ioan Salomie. Distributed ledger technology review and decentralized applications development guidelines. *Future Internet*, 13(3):62, 2021.

[8] Omar Alfandi, Salam Khanji, Liza Ahmad, and Asad Khattak. A survey on boosting iot security and privacy through blockchain: Exploration, requirements, and open issues. *Cluster Computing*, 24:37–55, 2021.

[9] Ahmed Afif Monrat, Olov Schelén, and Karl Andersson. A survey of blockchain from the perspectives of applications, challenges, and opportunities. *IEEE Access*, 7:117134–117151, 2019.

[10] Çetin Kaya Koç, Funda Özdemir, Zeynep Ödemiş Özger, Çetin Kaya Koç, Funda Özdemir, and Zeynep Ödemiş Özger. Rivest-shamir-adleman algorithm. *Partially Homomorphic Encryption*, pages 37–41, 2021.

[11] Maiya Al Saadi, Oman Muscat, and Basant Kumar. A review on elliptic curve cryptography. *International Journal of Future Generation Communication and Networking*, 13(3):1597–1601, 2020.

[12] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted execution environment: what it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/Ispa*, volume 1, pages 57–64. IEEE, 2015.

[13] Stathis Mavrovouniotis and Mick Ganley. Hardware security modules. In *Secure Smart Embedded Devices, Platforms and Applications*, pages 383–405. Springer, 2013.

[14] Antonio J Cabrera-Gutiérrez, Encarnación Castillo, Antonio Escobar-Molero, José A Álvarez-Bermejo, Diego P Morales, and Luis Parrilla. Integration of hardware security modules and permissioned blockchain in industrial iot networks. *IEEE Access*, 10:114331–114345, 2022.

[15] Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf. Trusted execution environments: properties, applications, and challenges. *IEEE Security & Privacy*, 18(2):56–60, 2020.

[16] Francisco Eugenio Potestad-Ordóñez, Erica Tena-Sánchez, Antonio José Acosta-Jiménez, Carlos Jesús Jiménez-Fernández, and Ricardo Chaves. Hardware countermeasures benchmarking against fault attacks. *Applied Sciences*, 12(5):2443, 2022.

[17] Trusted firmware-m. https://tf-m-user-guide.trustedfirmware.org/.

[18] Infineon Technologies AG. Psoc 64 - secured mcu. https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/psoc-6-32-bit-arm-cortex-m4-mcu/psoc-64//.

[19] Aamir Iqbal, Mohammad Amir, Vinod Kumar, Aftab Alam, and Mohammad Umair. Integration of next generation iiot with blockchain for the development of smart industries. *Emerg. Sci. J*, 4:1–17, 2020.

[20] Davor Dujak and Domagoj Sajter. Blockchain applications in supply chain. *SMART supply network*, pages 21–46, 2019.

[21] David Allessie, Maciej Sobolewski, Lorenzino Vaccari, and Francesco Pignatelli. Blockchain for digital government. *Luxembourg: Publications Office of the European Union*, pages 8–10, 2019.

[22] Ayesha Shahnaz, Usman Qamar, and Ayesha Khalid. Using blockchain for electronic health records. *IEEE access*, 7:147782–147795, 2019.

[23] Ye Guo and Chen Liang. Blockchain application and outlook in the banking industry. *Financial innovation*, 2:1–12, 2016.

[24] Giulio Caldarelli. Understanding the blockchain oracle problem: A call for action. *Information*, 11(11):509, 2020.

[25] Orlando Arias, Jacob Wurm, Khoa Hoang, and Yier Jin. Privacy and security in internet of things and wearable devices. *IEEE Transactions on Multi-Scale Computing Systems*, 1(2):99–109, 2015.

[26] Jasleen Kaur, Alka Agrawal, and Raees Ahmad Khan. Security issues in fog environment: a systematic literature review. *International Journal of Wireless Information Networks*, 27:467–483, 2020.

[27] Man-Ki Yoon, Sibin Mohan, Jaesik Choi, Jung-Eun Kim, and Lui Sha. Securecore: A multicore-based intrusion detection architecture for real-time embedded systems. In *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 21–32. IEEE, 2013.

[28] Mehryar Rahmatian, Hessam Kooti, Ian G Harris, and Elaheh Bozorgzadeh. Hardware-assisted detection of malicious software in embedded systems. *IEEE Embedded Systems Letters*, 4(4):94–97, 2012.

[29] Shufu Mao and Tilman Wolf. Hardware support for secure processing in embedded systems. In *Proceedings of the 44th annual Design Automation Conference*, pages 483–488, 2007.

[30] George Bravos, Antonio J Cabrera, Camilo Correa, Dragan Danilović, Nikolaos Evangeliou, Gilad Ezov, Zoran Gajica, Dušan Jakovetić, Leonidas Kallipolitis, Milan Lukić, et al. Cybersecurity for industrial internet of things: architecture, models and lessons learned. *IEEE Access*, 10:124747–124765, 2022.

[31] Zhao Huang and Quan Wang. A puf-based unified identity verification framework for secure iot hardware via device authentication. *World Wide Web*, 23(2):1057–1088, 2020.

[32] Kaspar Matas, Tuan La, Nikola Grunchevski, Khoa Pham, and Dirk Koch. Invited tutorial: Fpga hardware security for datacenters and beyond. In *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 11–20, 2020.

[33] Festus Hategekimana, Taylor JL Whitaker, Md Jubaer Hossain Pantho, and Christophe Bobda. Iot device security through dynamic hardware isolation with cloud-based update. *Journal of Systems Architecture*, 109:101827, 2020.

[34] Travis Peters, Reshma Lal, Srikanth Varadarajan, Pradeep Pappachan, and David Kotz. Bastion-sgx: Bluetooth and architectural support for trusted i/o on sgx.

In *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, pages 1–9, 2018.

[35] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, pages 1–9. 2016.

[36] Ning Zhang, Jin Li, Wenjing Lou, and Y Thomas Hou. Privacyguard: Enforcing private data usage with blockchain and attested execution. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings 13*, pages 345–353. Springer, 2018.

[37] Chen Cao, Le Guan, Ning Zhang, Neng Gao, Jingqiang Lin, Bo Luo, Peng Liu, Ji Xiang, and Wenjing Lou. Cryptme: Data leakage prevention for unmodified programs on arm devices. In *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21*, pages 380–400. Springer, 2018.

[38] Juan Wang, Zhi Hong, Yuhan Zhang, and Yier Jin. Enabling security-enhanced attestation with intel sgx for remote terminal and iot. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):88–96, 2017.

[39] Robert Pettersen, Håvard D Johansen, and Dag Johansen. Secure edge computing with arm trustzone. In *IoTBDS*, pages 102–109, 2017.

[40] Tiago Alves. Trustzone: Integrated hardware and software security. *Information Quarterly*, 3:18–24, 2004.

[41] Le Guan, Peng Liu, Xinyu Xing, Xinyang Ge, Shengzhi Zhang, Meng Yu, and Trent Jaeger. Trustshadow: Secure execution of unmodified applications with arm trustzone. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 488–501, 2017.

[42] Dylan Paulin, Christine Hennebert, Thibault Franco-Rondisson, Romain Jayles, Thomas Loubier, and Raphaël Collado. Histotrust: Ethereum-based attestation of a data history built with op-tee and tpm. In *International Symposium on Foundations and Practice of Security*, pages 130–145. Springer, 2021.

[43] Carlton Shepherd, Raja Naeem Akram, and Konstantinos Markantonakis. Emlog: tamper-resistant system logging for constrained devices with tees. In *Information Security Theory and Practice: 11th IFIP WG 11.2 International Conference, WISTP 2017, Heraklion, Crete, Greece, September 28–29, 2017, Proceedings 11*, pages 75–92. Springer, 2018.

[44] Infineon Technologies AG. S2go pressure dps310.

[45] Infineon Technologies AG. Optiga embedded security solutions.

[46] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63, 2001.

[47] Alexander Yakubov, Wazen Shbair, Anders Wallbom, David Sanda, et al. A blockchain-based pki management framework. In *The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018, Tapei, Tawain 23-27 April 2018*, 2018.

[48] Shahinaz K Ezzat, Yasmine NM Saleh, and Ayman A Abdel-Hamid. Blockchain oracles: State-of-the-art and research directions. *IEEE Access*, 2022.

[49] Giulio Caldarelli. Overview of blockchain oracle research. *Future Internet*, 14(6):175, 2022.

[50] Amirmohammad Pasdar, Zhongli Dong, and Young Choon Lee. Blockchain oracle design patterns. *arXiv preprint arXiv:2106.09349*, 2021.

[51] Roman Mühlberger, Stefan Bachhofner, Eduardo Castelló Ferrer, Claudio Di Ciccio, Ingo Weber, Maximilian Wöhrer, and Uwe Zdun. Foundational oracle patterns: Connecting blockchain to the off-chain world. In *Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2020 Blockchain and RPA Forum, Seville, Spain, September 13–18, 2020, Proceedings 18*, pages 35–51. Springer, 2020.

[52] Zahra Kazemi, Athanasios Papadimitriou, Ioanna Souvatzoglou, Ehsan Aerabi, Mosabbah Mushir Ahmed, David Hely, and Vincent Beroulle. On a low cost fault injection framework for security assessment of cyber-physical systems: Clock glitch attacks. In *2019 IEEE 4th International Verification and Security Workshop (IVSW)*, pages 7–12. IEEE, 2019.

[53] David Mills, Jim Martin, Jack Burbank, and William Kasch. Network time protocol version 4: Protocol and algorithms specification. Technical report, 2010.

[54] Daniel Fox Franke, Dieter Sibold, Kristof Teichel, Marcus Dansarie, and Ragnar Sundblad. Network time security for the network time protocol, 2020.

[55] Gabriel Estevam, Lucas M Palma, Luan R Silva, Jean E Martina, and Martin Vigil. Accurate and decentralized timestamping using smart contracts on the ethereum blockchain. *Information Processing & Management*, 58(3):102471, 2021.

[56] Javier Echave, Marta Barral, Maria Fraga-Corral, Miguel A Prieto, and Jesus Simal-Gandara. Bottle aging and storage of wines: A review. *Molecules*, 26(3):713, 2021.

[57] CS Ough. Some effects of temperature and so2 on wine during simulated transport or storage. *American Journal of Enology and Viticulture*, 36(1):18–22, 1985.

[58] Edvard Tijan, Saša Aksentijević, Katarina Ivanić, and Mladen Jardas. Blockchain technology implementation in logistics. *Sustainability*, 11(4):1185, 2019.

[59] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

Part III

Conference publications

7

Integration of Hardware Security Modules in Blockchain Networks

ANTONIO J. CABRERA-GUTIERREZ[1,2], ENCARNACION CASTILLO[2], ANTONIO ESCOBAR-MOLERO[1], JOSE A. ALVAREZ-BERMEJO[3], DIEGO P. MORALES[2], and LUIS PARRILLA[2].

1. Infineon Technologies AG, Am Campeon 1-15, 85579, Neubiberg, Bavaria, Germany
2. University of Granada, Department of Electronics and Computer Technology, Faculty of Sciences, Granada, 18071, Andalucia, Spain
3. University of Almeria, Department of Informatics, Carr. Sacramento s/n, Almeria, 04120, Andalucia, Spain

Conference: 22th International Conference Computational and Mathematical Methods in Science and Engineering

Abstract

Hardware Security Modules (HSM) are used as a hardware based root of trust that offers physical protection to the architecture where they are applied. When HSM is combined with decentralized ledger technologies as Blockchain, they offer security by horizontal device-to-device communication and hardware-enabled security in the device where the HSM is placed. This work presents a study on the integration of these two technologies, shedding light on the elements that these two technologies should have in common in order to be compatible, which are cryptographic curves and cryptographic standards, and highlighting the benefits they bring to the architecture where they are applied.

7.1 Introduction

In the era of digitalization, security threats are one of the main problems to be solved. To address this problem, new technologies and paradigms have emerged in the past few years, being Blockchain networks one of the more promising tools for that, starting to be used in a wide range of applications in different fields [1] [2] [3] [4]. Not as trendy, but not less important are Hardware Security Modules (HSMs), which are becoming fundamental elements for all types of hardware architectures to fight against the rising of cybercrime in last years [5].

Blockchain networks are based on decentralized ledger technologies (DLT) [6], which enables secure functioning of a decentralized database. DLT allows storage of data in a secure way using cryptography and the fact that its decentralisation prevents manipulation of the data stored in the database. Blockchain networks are based on distributed copies of a database including the transactions made by the network participants. Furthermore, these kinds of networks have a consensus algorithm previously agreed by all the members of the network, thus allowing authenticity and immutability of those records.

What makes Blockchain networks different is the underlying cryptographic framework, which enables device-to-device communications security in these networks. Blockchain networks use cryptographic material such as public/private cryptosystems and certificates. In addition, Blockchain networks make intensive use of cryptographic operations such as signing, signature verification or key generation. Typically, the keys and certificates involved in the Blockchain are stored in a "software wallet" [7].

Software alone is not enough to protect cryptographic keys and certificates as the stored data can be read, modified and distributed effortlessly [8]. In order to avoid an easy access and manipulation to this cryptographic material, a hardware-based security becomes necessary. HSMs [9] offer a solution which relies on:

- High entropy key generation.
- Tamper-proof protection, enabling secure storage of private keys or sensitive information.
- Key back-up and restoration.

These features offered by HSMs make them ideal to be used with Blockchain networks, as keys can be stored securely and can even be generated on these devices via a True Random Number Generator (TRNG), making them much more difficult to replicate. HSMs present tamper resistance which avoids physicals attacks such as probing attacks. An HSM is placed next to a host (CPU or MCU) which asks to the HSM to perform

different security tasks such as: signing, signature validation, encryption, decryption or hashing.

In combination with Blockchain, the intensive use of cryptographic operations in these networks is carried out on the HSM instead of on the host, freeing up computational load on the host in addition to offer a hardware security layer over the one already offered by Blockchain.

7.2 General description

Current security paradigms on computer networks are based on Public Key Infrastructure (PKI) shown in Fig. 1.

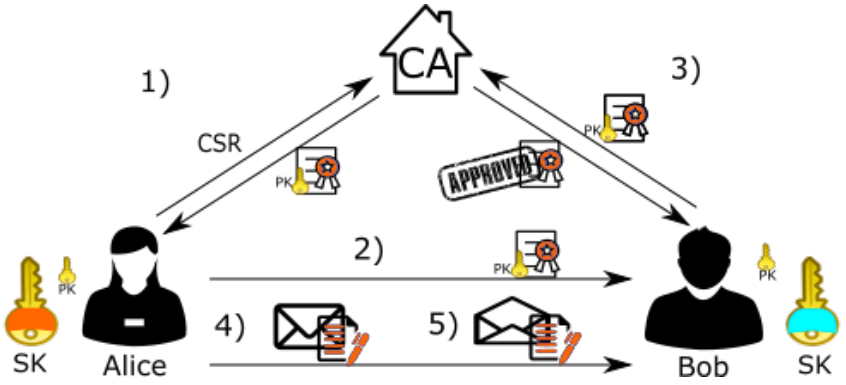

FIGURE 7.1: Public Key Infrastructure

In this type of architectures, the security relies on a Certification Authority (CA) which sometimes splits in Validation Authority (VA) and in Registration Authority (RA) [10]. This presents several problems, the main one is that there is only a single point of failure. Attacks against the CA can have a major impact since it is the entity that provides identity and authenticity to all users in the network.

With the rise of new technologies and new paradigms such as IoT, PKI architectures become obsolete or at least quite vulnerable. Storing sensitive or critical data in a central point of the network becomes a very attractive vulnerability for attackers, who simply by accessing this central entity are able to modify the system to their own way.

The solution to such architectures lies in decentralising these central entities as shown in Fig. 2.

In a PKI architecture, Alice must authenticate with the CA (Step 1), she sends a Certificate Signing Request (CSR) and receives her certificate. She exchanges her certificate with Bob (Step 2) before sending the message and he verifies it again with the CA (Step 3). Once this is done, all messages sent by Alice (Step 4) can be decrypted by Bob (Step 5) as he has her certificate and it has been previously verified.

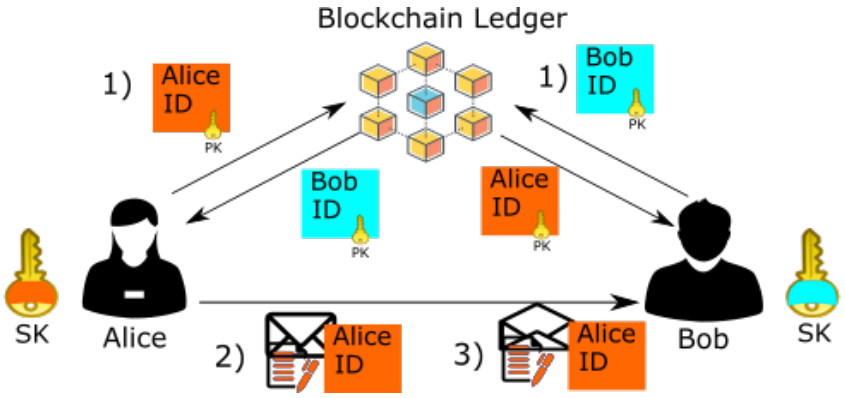

FIGURE 7.2: Decentralized PKI

In a decentralised PKI, the CA is replaced by a decentralised database (Blockchain network), which is owned by all users of the network as they keep a copy and are responsible for validating the transactions carried out on the network. In this case, both Alice and Bob are registered in the Blockchain ledger by sending an ID with their public key (Step 1). In the ledger, there will be a copy of all registered users with their ID and public key. Bob can find out Alice's public key by querying the ledger since each user has a copy of it. In this way, messages signed by Alice are identified with the ID (Step 2) given to her by the Blockchain and Bob can decrypt them using the public key obtained in the ledger (Step 3).

Decentralising the network using Blockchain technologies can mitigate different types of attacks on the architecture such as Denial of Service (DoS) attacks. However, if the attacker is able to obtain the user's keys or certificates, the system can be broken easily. In this case HSMs have an important advantage to maintain the protection of the cryptographic material such as private keys.

The integration of these two technologies becomes critical to maintaining the integrity of the system. In order to make such technologies compatible with each other, the following specifications have to be met:

- Cryptographic standards: Public Key Cryptographic Standards (PCKS) establish the security requirements, protocols and algorithms to prevent security weaknesses and backdoors. There are several PKCSs used in communications where HSMs are involved, for example PKCS10 [11], also known as a Certificate Signing Request (CSR), specifies the format of messages to and from a CA. Once the CA issues a certificate in X.509 format it is stored in the HSM. On the other hand, PKCS11 [12] defines access methods to cryptographic tokens such as HSMs, smart cards and others. This standard isolates the application from the cryptographic device. In this sense, the PKCS11 standard works as an interface between HSMs and the Blockchain network and becomes a key element in enabling the interaction between these two technologies.

- Elliptic Cryptographic Curves (ECC): As explained above, HSMs are able to generate keys within them because they incorporate TRNGs. These modules generate random numbers with high entropy due to external physical phenomena, making it almost impossible for a key generated with a TRNG to be replicable. These generated keys have properties that make them satisfy the mathematical equations of the elliptic curve for which they have been generated. There are a wide variety of ECC and HSMs that usually support NIST standard ECs [13] as prime256v1, and also secp256k1. These include the secp256k1 which is the one used in Bitcoin or Ethereum. Since the cryptographic operations used in Blockchain are performed in the HSM, these two technologies must support the same cryptographic curves and algorithms, making this a mandatory property when integrating these two elements.

When both a Blockchain network and HSMs share the same cryptographic standards and elliptic curve, their joint application is ideal. Thanks to the rise of Blockchain technologies and certain applications such as cryptocurrencies, more and more HSMs are using these types of standards and curves. There are even groups such as the Trusted Computing Group (TCG) that are dedicated to developing these types of standards.

7.3 Conclusions

Thanks to technologies such as Bitcoin or Ethereum, Blockchain networks are starting to be used in other fields such as energy, inbound logistics or automotive and mobility. The security of these new application fields for Blockchain networks have to increase. In this sense and to ensure a complete security, the use of a hardware device that helps to protect the cryptographic elements from external attackers will be required. This is where HSMs have their role to play.

The increasing deployment of these technologies in industrial world and the increasing standardisation of the technologies related to them (cryptographic curves and cryptographic standards) mean that these technologies are bound to go hand in hand in the future.

HSMs and Blockchain complement each other perfectly as each of these technologies offers different levels of security. On the one hand, at the hardware level and, on the other hand, at the level of communication between different users, which makes them a perfect combination.

7.4 Acknowledgments

This work was supported by Infineon Technologies AG. Additional funding was provided by European Union's Horizon 2020 research and innovation program through the funding project Cyber security 4.0: protecting the Industrial Internet of Things (C4IIoT) under the grant agreement No 833828. The work has also been funded by Consejería de Economía y conocimiento Junta de Andalucía (Spain) and European Regional Development Funds (ERDF) under project B-TIC-588-UGR20.

7.5 References

[1] Pradip Kumar Sharma, Neeraj Kumar, and Jong Hyuk Park. Blockchain-based distributed framework for automotive industry in a smart city. *IEEE Transactions on Industrial Informatics*, 15(7):4197–4205, 2018.

[2] Şeref Bülbül and Gökhan İnce. Blockchain-based framework for customer loyalty program. In *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, pages 342–346. IEEE, 2018.

[3] Guido Perboli, Stefano Musso, and Mariangela Rosano. Blockchain in logistics and supply chain: A lean approach for designing real-world use cases. *Ieee Access*, 6:62018–62028, 2018.

[4] Se-Chang Oh, Min-Soo Kim, Yoon Park, Gyu-Tak Roh, and Chin-Woo Lee. Implementation of blockchain-based energy trading system. *Asia Pacific Journal of Innovation and Entrepreneurship*, 2017.

[5] George Loukas. *Cyber-physical attacks: A growing invisible threat.* Butterworth-Heinemann, 2015.

[6] David C Mills, Kathy Wang, Brendan Malone, Anjana Ravi, Jeffrey Marquardt, Anton I Badev, Timothy Brezinski, Linda Fahy, Kimberley Liao, Vanessa Kargenian, et al. Distributed ledger technology in payments, clearing, and settlement. 2016.

[7] Akanksha Kaushik, Archana Choudhary, Chinmay Ektare, Deepti Thomas, and Syed Akram. Blockchain—literature survey. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 2145–2148. IEEE, 2017.

[8] Nate Lawson. Side-channel attacks on cryptographic software. *IEEE Security & Privacy*, 7(6):65–68, 2009.

[9] Stathis Mavrovouniotis and Mick Ganley. Hardware security modules. In *Secure Smart Embedded Devices, Platforms and Applications*, pages 383–405. Springer, 2013.

[10] Zhihua Li, Xi Yin, Zhenmin Geng, Haitao Zhang, Pengfei Li, Ya Sun, Huawei Zhang, and Lin Li. Research on pki-like protocol for the internet of things. In *2013 Fifth International Conference on Measuring Technology and Mechatronics Automation*, pages 915–918. IEEE, 2013.

[11] Magnus Nystrom and Burt Kaliski. Pkcs# 10: Certification request syntax specification version 1.7. Technical report, 2000.

[12] OASIS Standard. Pkcs# 11 cryptographic token interface base specification version 2.40. *Recuperado de https://bit. ly/3EJ18jk [Consultado noviembre 2021]*, 2015.

[13] Cameron F Kerry and Patrick D Gallagher. Digital signature standard (dss). *FIPS PUB*, pages 186–4, 2013.

8

Blockchain-based implementation of Tradable Green Certificates

ANTONIO J. CABRERA-GUTIERREZ[1,2], ENCARNACION CASTILLO[2], ANTONIO ESCOBAR-MOLERO[1], DIEGO P. MORALES[2], and LUIS PARRILLA[2].

1. Infineon Technologies AG, Am Campeon 1-15, 85579, Neubiberg, Bavaria, Germany
2. University of Granada, Department of Electronics and Computer Technology, Faculty of Sciences, Granada, 18071, Andalucia, Spain

Conference: 18th International Conference on PhD Research in Microelectronics and Electronics

Abstract

Blockchain networks, given that they are not based on power-hungry proof-of-work methods, can be used in smart grid applications, in particular in the tradable green certificates use case. Thanks to the Blockchain and the smart contracts implemented, certificates can be tracked and exchanged between entities without the intervention of third parties, as well as keeping an immutable and reliable record of these certificates. This work proposes a hardware implementation of a green certificate trading system based on Blockchain, in which prosumers use hardware secure elements to implement the cryptographic tools used to interact with the Blockchain. Smart contracts help to automate these processes, deleting intermediaries, saving costs and avoiding bureaucracy.

8.1 Introduction

Decarbonisation is leading to increase the use of renewable energies, thus turning out to a diversity of energy prosumers types. The result is a large decentralisation of energy production, which has traditionally been centralised in large electricity production plants [1]. This decentralisation is reflected in the smart grids and micro grids currently emerging in the energy sector. To increase this trend and motivate renewable energy adoption, different measures are being taken by governmental entities (e. g. in 2021 the EU launched the new Corporate Sustainability Reporting Directive [2]).

This transformation of the energy market creates new challenges, such as tracking the origin of the electricity, since in a wide market, it is necessary to know who is the energy source, what type of energy source it uses and whether it is green or not [3]. To address these matters, Tradable Green Certificates (TGCs) and Renewable Energy Certificates (RECs) schemes are implemented in this kind of environments. A TGC or REC represents the generation of a certain amount of electricity from a renewable energy source. Market players can sell TGCs to gain profits, thus promoting the consumption of renewable energy by market players who have the responsibility of utilising renewable energy [4].

To help the implementation of these certificates, technologies such as Blockchain can help. Through properties as decentralisation, transparency and security, Blockchain offers special features that help to promote renewable energies into the current energy market [5]. Blockchain implements smart contracts that are capable of generating, tracking and exchanging these certificates without the need of intermediaries and keep a reliable and immutable record of the transaction between the entities which compound the grid. In addition, the necessary reward mechanisms can be implemented for these certificates [6], e.g. electricity price reductions or as a proof of compliance with green regulations.

This article proposes the use of Blockchain to solve the problem of energy tracking by generating TGCs as proof of this. Through a set up composed of different hardware nodes simulating prosumers, a simulation is carried out in which these prosumers exchange TGC through the Blockchain. Each node is identified by a Digital Identity (DI) generated through a Hardware Security Module (HSM). The nodes run a Blockchain client which delegates all cryptographic operations to the HSM. In this way, a new layer of security is obtained at the hardware level, which makes the Blockchain implemented on these nodes more robust and secure.

The rest of the article is arranged as following: Section II reviews the related literature and works; in Section III the design of the architecture is explained focusing on the Blockchain used (Hyperledger Fabric, HLF) and the HSM; in Section IV the implementation of the whole system is illustrated, and finally, Section V gathers the conclusions.

8.2 State of the art

The idea of implement TGCs in smart grids based on Blockchain has been emerging over the past few years. It is true that the foundation of these certificates bears certain similarities to cryptocurrencies –a pioneering application of Blockchain technologies–, which has led to the description of different projects and proposals in relation to these certificates treated as monetisable tokens [7].

Proposals as SolarCoin [8] describes a system where the users can produce solar energy and send it to the smart grid receiving 1 SolarCoin per 1 MWh of validated electricity production. The money can be exchanged with other cryptocurrencies or traditional currencies. In NRGCoin [9], the smart contracts running on Ethereum enable trading between producers and consumers avoiding intermediaries as well as inflation since the cryptocurrencies value remains the same as far as the amount of energy is traded. One NRGCoin will always be 1-kWh. EECoin [10] is an energy buying and selling platform adding compatibility to ERC20 Ethereum token standard.

On the other hand, a framework for developing RECs based on cryptocurrencies on Ethereum is proposed in [11]. The work described in [12] explores the use of Blockchain technology for TGC use case, and it is demonstrated in several solar energy-powered buildings. The authors of [13] analyse two use cases of application of Blockchain to energy certificates. In [14], the authors simulate a marketplace using the Ethereum Blockchain and smart contracts, where prosumers can sell tokenised certificates to consumers willing to subsidise renewable energy producers. The work described in [15] establishes a comparison between two energy markets, TGC and tradable white certificates markets. The authors analyse how to coordinate these two energy markets and what would be the main issues under a reward-penalty mechanism.

These works are among the few in the literature on the use of Blockchain in energy trading based on certificates, as most of the literature is based on exchanging amounts of energy [16]. The main step forward that this work brings over the previous ones is the simulation of certificate exchange implemented in hardware nodes using hardware secure element for signing and key generation in the operations of the Blockchain.

8.3 Architecture design

Typically, energy exchange in a smart grid using Blockchain is carried out by smart contracts where amounts of energy are sold and bought from different prosumers. However, the introduction of TGC brings an added value to these exchanges. Although a TGC can be equivalent to a certain amount of energy, it also brings information as the proof that the energy comes from a renewable source. This information as well as other like

DI, company, type of renewable energy source, date of issuing, etc., can be stored in the Blockchain.

8.3.1 Blockchain architecture

The design system proposed is built using permissioned Blockchain networks as HLF [17]. Permissioned Blockchain is a distributed ledger which is not publicly accessible. The participation of a member in the network requires certain permissions granted at registration time by Blockchain administrators through certificates offering additional security layer over traditional Blockchain networks. Since this kind of Blockchains do not use Proof of Work, they avoid power-hungry processing [18].

In HLF, prosumers act as clients of the network in which they interact with each other. Prosumers can belong to different organisations through which they are admitted into the Blockchain, while the Blockchain is formed by a consortium of companies or individuals.

Figure 1 shows the outline of the proposed design in which two prosumers (*P1* and *P2*) from different organisations communicate through the channel *C1*, after they have been registered within the network through the certification authority (CA) respective to their organisation (*CA1* and *CA2*).

The network is maintained by the Orderer (O) who belongs to an independent organisation and sets the channel policies (CPs) agreed by the two organisations to which the prosumers belong (RA and RB). Each prosumer has two peer nodes ($P1_1$ and $P2_1$, and $P1_2$ and $P2_2$, respectively) that are responsible for maintaining the ledger (L) and executing the smart contracts (SCs).

8.3.2 Hardware security

The prosumers are in charge of sending transactions to the Blockchain network. These transactions are evaluated and validated by the peers nodes, but before being validated, these transactions are signed by the HSM's prosumers.

This signature mechanism is based on Elliptic Curve Digital Signature Algorithm (ECDSA) [19] which uses Elliptic Cryptographic Curve (ECC) keys for signature. In particular, in this proposal the curve is the secp256r1 also known as NIST P-256.

Typically, the software implementations of ECC keys generation process and signature mechanisms show security vulnerabilities, as the key can be read, modified and distributed effortlessly [20]. For these reasons, we propose to perform these operations using an HSM attached to the node where the prosumer is simulated.

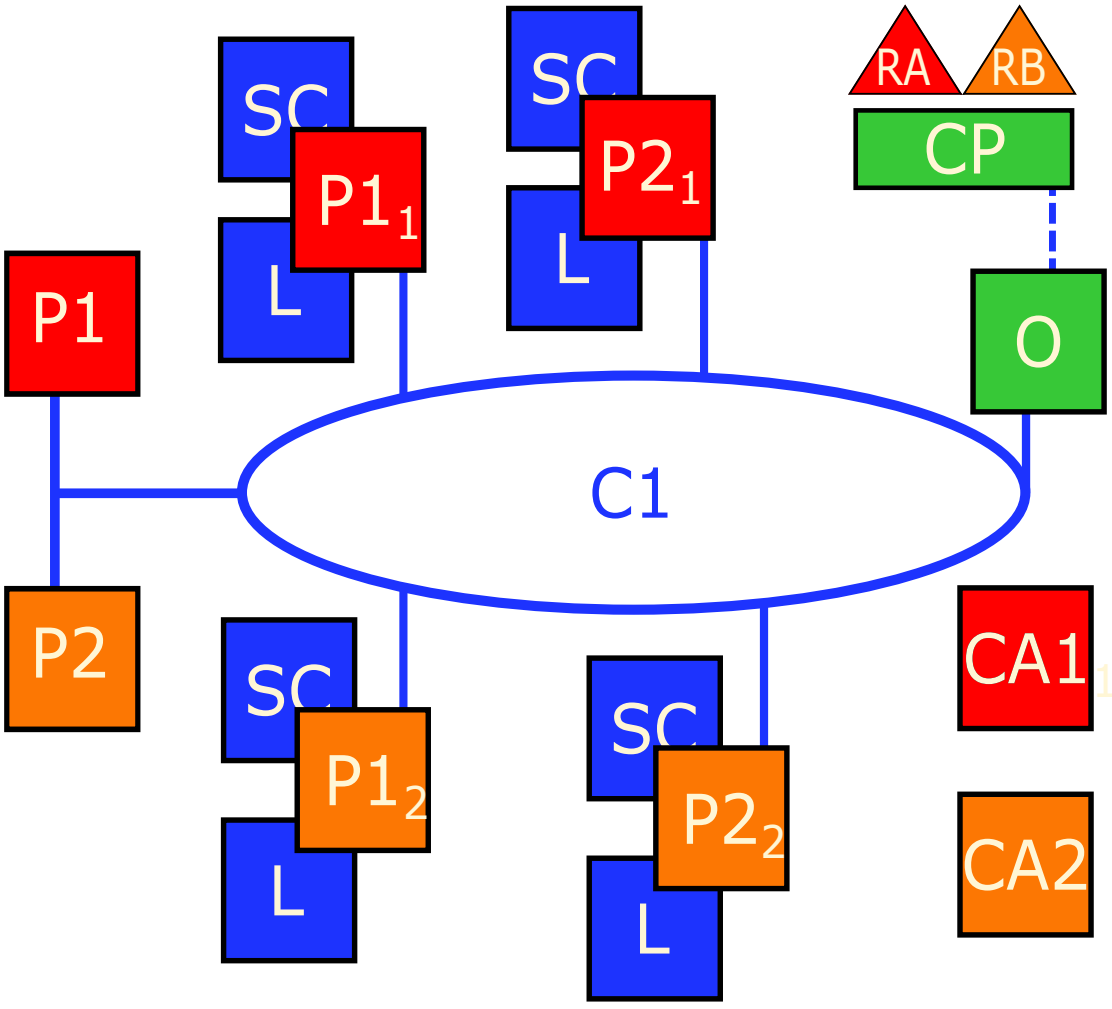

FIGURE 8.1: HLF architecture.

The HSM is also equipped with a True Random Number Generator (TRNG) for enhancing the security strength of the key generation process. Such a unit generates true random numbers from high-entropy microscopic physical events, enabling the HSM to offer certified protection against side-channel attacks.

Both signature and key generation mechanisms are used by the prosumer in the enrolment process and invoking smart contract functions. Figure 2 shows how the prosumers generate a key pair in the HSM before sending a Certificate Signing Request (CSR) to the CA in order to get an X.509 certificate for interacting later with the Blockchain. Also, in Figure 2, the prosumer sends the hash of the transaction that invokes a smart contract function to the HSM for signing with the private key.

These mechanisms form the basis on which the logic implemented in smart contracts is built, and which is used to create and exchange TGCs as well as create the DIs of the prosumers.

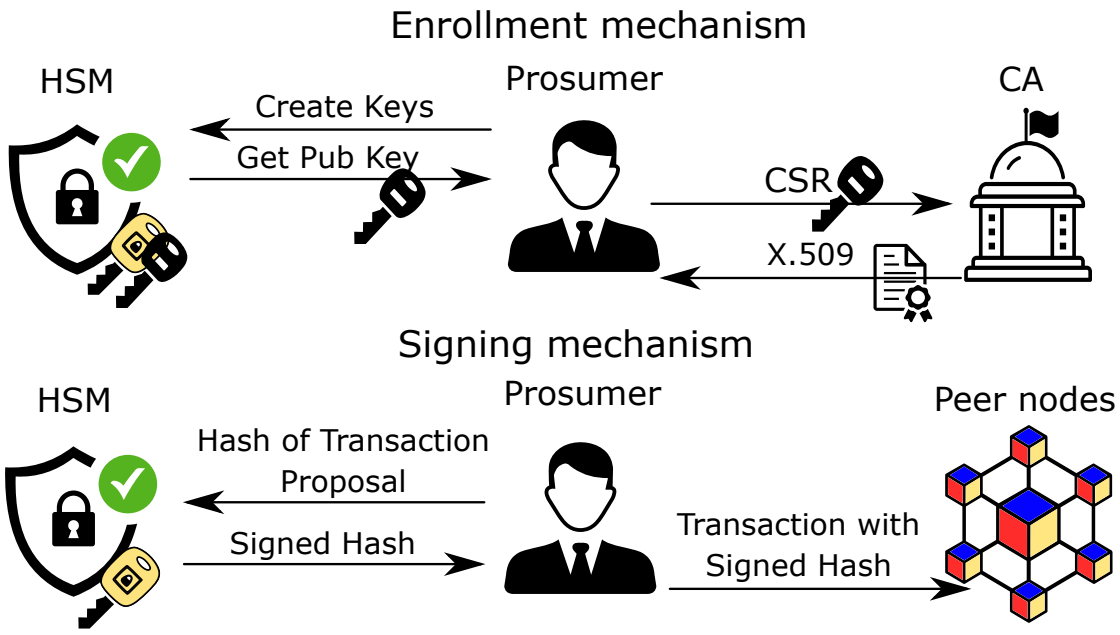

FIGURE 8.2: Mechanisms where HSM is involved.

8.4 Implementation

The architecture designed and described above has been deployed on a computer where each Blockchain entity is a virtualized container, each prosumers have been deployed on a Raspberry Pi (RPI), to which an HSM has been connected to perform the necessary cryptographic operations, as it is shown in Figure 3.

Prosumers generate through the HSM and the public key infrastructure a DI that is stored in the Blockchain. This is compound of several fields (includes a unique string identifier) inside of a certificate which identifies the device and will function as a fingerprint for tracking TGCs.

Inside the set up, Figure 4, one of the prosumers has a solar panel, connected to it, which produces energy. Through the logic implemented in smart contracts, when the consumer starts to consume energy; the producer, depending on the amount of energy produced and stored, will generate a certain amount of TGCs that are sent to the consumer. In the event that the amount of energy stored by the producer does not reach the amount necessary to satisfy the needs, the energy used will be non-green energy and the consumer will not receive TGCs during that time.

In this way, the Blockchain ledger will keep a count of all the TGCs sent, which will be identified with a timestamp, the DI of the origin and the DI of the destination. Each node, identified by their DI on the Blockchain, has a count of the number of TGCs produced or the number of TGCs obtained by consuming green energy. Table 1 shows this information contained in the ledger related to the nodes. This table shows that the number of consumer TGCs coincides in absolute value with the number of TGCs

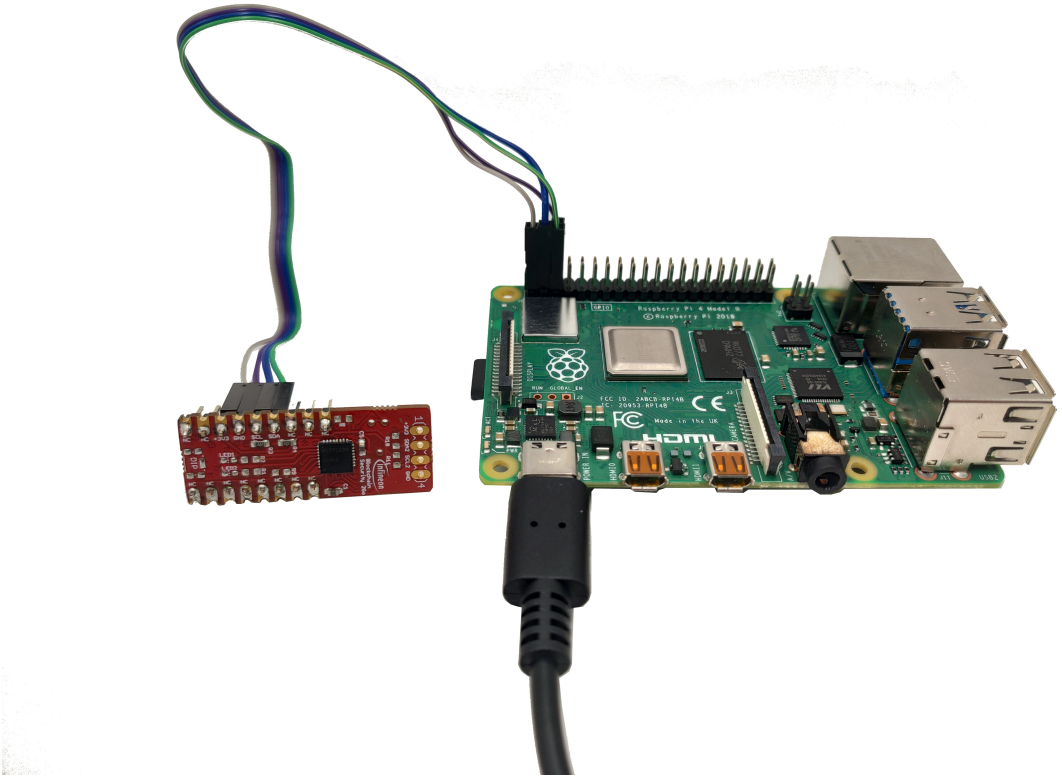

FIGURE 8.3: RPI with HSM

issued by the producer. This is because all the TGCs issued by the producer have been consumed by the consumer node.

TABLE 8.1: Information keeping in the ledger related to the nodes.

	Attributes		
	N° TGC	**Role**	**Digital Identity**
Node 1	-20	producer	CwsY...ASs
Node 2	20	consumer	kXzv...nfS

In Table 2, the information regarding the transaction of the TGC is gather. This information shows three TGCs identified with the timestamp, the signer and the receiver. As it can be observed, the signer for the three TGCs is the same, as well as the receiver for the three TGCs. This is again due to the network is made up of two entities in this example, one acting as signer (producer) and the other one as receiver (consumer).

TABLE 8.2: Information related transaction of TGCs.

	Timestamp	**Signer**	**Receiver**
TGC N° 1	1675877934	CwsY...ASs	kXzv...nfS
TGC N° 2	1675888943	CwsY...ASs	kXzv...nfS
TGC N° 3	1675905425	CwsY...ASs	kXzv...nfS

Finally, Figure 5 shows an example of the interaction flow between the two entities.

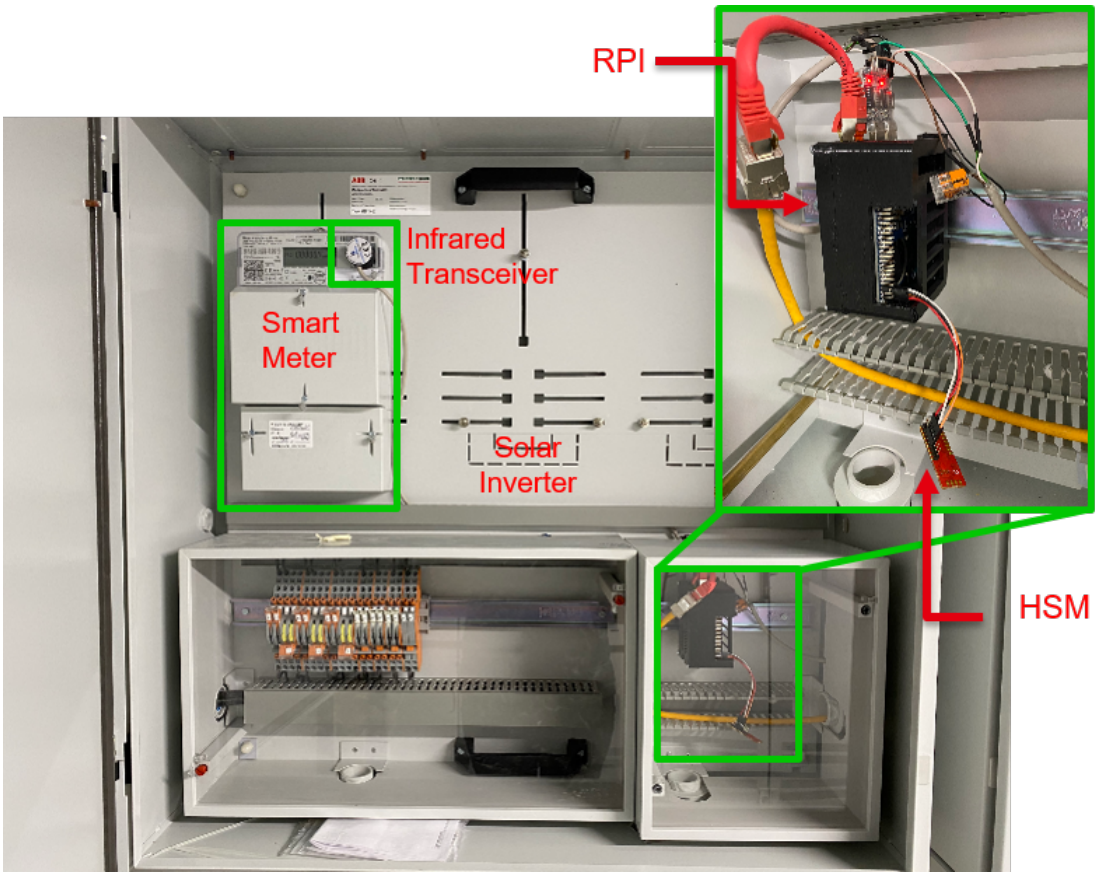

FIGURE 8.4: Hardware architecture demonstrator.

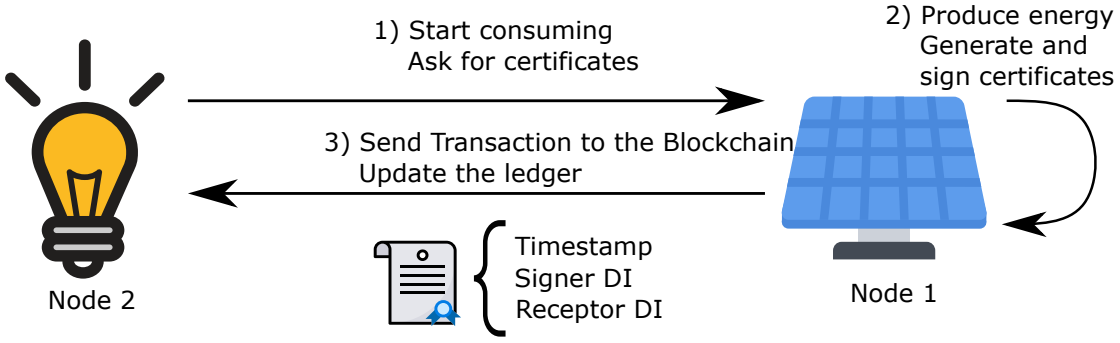

FIGURE 8.5: Interaction between producer and consumer

In this way, following this interaction flow and through the presented hardware, the two nodes interact with each other producing TGC and using them to consume energy, keeping all the records of TGCs transaction in the Blockchain. All these interactions are done and automated through smart contracts. This demonstrator shows the feasibility of this concept and could be extended to a real-life scenario, for example in Trusted Green Charging project [21], where we deployed a similar architecture.

8.5 Conclusions

The use of TGC is increasing the deployment of renewable energies as well as the interaction of these energies with non-clean energies. Through technologies such as Blockchain, the implementation of these TGCs is increasing. From the deployment of Blockchain networks such as HLF in which prosumers are mapped to exchange this type of certificates, this use case is automated through smart contracts.

In the implementation proposed in this paper, this use case is secured through the use of HSMs in the interaction with the Blockchain. Using this type of devices, all cryptographic operations are performed on a device that offers resistance to physical attacks adding a new security layer to the architecture with respect to the traditional software implementations. Through the HSMs, the signing of transactions on the Blockchain, as well as the registration of prosumers on the Blockchain are carried out. DIs are also used in this implementation for the identification of prosumers on the Blockchain.

The whole logic of the exchange of the TGCs is automated through smart contracts. In this way, and with the presented hardware deployment, a TGC system is simulated in which no external entities interfere, a well-protected and reliable record of TGCs exchanges is kept whilst costs are saved avoiding bureaucracy and extra processes.

8.6 Acknowledgements

This work was supported by Infineon Technologies AG. Additional funding was received from the Electronic Components and Systems for European Leadership Joint Undertaking under grant agreement No 876868 (PROGRESSUS). This Joint Undertaking receives support from the European Union‘s Horizon 2020 research and innovation programme and Germany, Slovakia, Netherlands, Spain, Italy. The work has also been funded by FEDER/Junta de Andalucia-Consejeria de Transformacion Economica, Industria, Conocimiento y Universidades under Project B-TIC-588-UGR20.

8.7 References

[1] Elisa Papadis and George Tsatsaronis. Challenges in the decarbonization of the energy sector. *Energy*, 205:118025, 2020.

[2] SPINACI STEFANO. Corporate sustainability reporting directive. 2022.

[3] Clemens Gerbaulet, Christian von Hirschhausen, Claudia Kemfert, Casimir Lorenz, and P-Y Oei. European electricity sector decarbonization under different levels of foresight. *Renewable energy*, 141:973–987, 2019.

[4] Fangyuan Zhao, Xin Guo, and Wai Kin Chan. Individual green certificates on blockchain: A simulation approach. *Sustainability*, 12(9):3942, 2020.

[5] Naiyu Wang, Xiao Zhou, Xin Lu, Zhitao Guan, Longfei Wu, Xiaojiang Du, and Mohsen Guizani. When energy trading meets blockchain in electrical power system: The state of the art. *Applied Sciences*, 9(8):1561, 2019.

[6] Fei Teng, Qi Zhang, Ge Wang, Jiangfeng Liu, and Hailong Li. A comprehensive review of energy blockchain: Application scenarios and development trends. *International Journal of Energy Research*, 45(12):17515–17531, 2021.

[7] Jiabin Bao, Debiao He, Min Luo, and Kim-Kwang Raymond Choo. A survey of blockchain applications in the energy sector. *IEEE Systems Journal*, 15(3):3370–3381, 2020.

[8] A Ekblaw and A Lippman. Solarcoin: Making blockchain mining open, green, and ready for micropayments. *PubPub. Document Version 1.0*, 2016.

[9] Mihail Mihaylov, Sergio Jurado, Narcís Avellana, Kristof Van Moffaert, Ildefons Magrans de Abril, and Ann Nowé. Nrgcoin: Virtual currency for trading of renewable energy in smart grids. In *11th International conference on the European energy market (EEM14)*, pages 1–6. IEEE, 2014.

[10] Jason Dispenza, Christina Garcia, and Ryan Molecke. Energy efficiency coin (eecoin) a blockchain asset class pegged to renewable energy markets. *EnLedger Corp., Lewes, DE, USA, Tech. Rep*, 1, 2017.

[11] R. Leonhard. *Developing Renewable Energy Credits as Cryptocurrency on Ethereum's Blockchain.*

[12] F. Imbault, M. Swiatek, R. de Beaufort, and R. Plana. The green blockchain: Managing decentralized energy production and consumption. In *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC / ICPS Europe)*, pages 1–5, 2017.

[13] Asma Khatoon, Piyush Verma, Jo Southernwood, Beth Massey, and Peter Corcoran. Blockchain in energy efficiency: Potential applications and benefits. *Energies*, 12(17):3317, 2019.

[14] J Alejandro F Castellanos, Debora Coll-Mayor, and José Antonio Notholt. Cryptocurrency as guarantees of origin: Simulating a green certificate market with the ethereum blockchain. In *2017 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, pages 367–372. IEEE, 2017.

[15] Soroush Safarzadeh, Ashkan Hafezalkotob, and Hamed Jafari. Energy supply chain empowerment through tradable green and white certificates: A pathway to sustainable energy generation. *Applied Energy*, 323:119601, 2022.

[16] Naiyu Wang, Xiao Zhou, Xin Lu, Zhitao Guan, Longfei Wu, Xiaojiang Du, and Mohsen Guizani. When energy trading meets blockchain in electrical power system: The state of the art. *Applied Sciences*, 9(8):1561, 2019.

[17] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[18] Mohammad Dabbagh, Mohsen Kakavand, Mohammad Tahir, and Angela Amphawan. Performance analysis of blockchain platforms: Empirical evaluation of hyperledger fabric and ethereum. In *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, pages 1–6. IEEE, 2020.

[19] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63, 2001.

[20] Antonio J Cabrera-Gutiérrez, Encarnación Castillo, Antonio Escobar-Molero, José A Álvarez-Bermejo, Diego P Morales, and Luis Parrilla. Integration of hardware security modules and permissioned blockchain in industrial iot networks. *IEEE Access*, 10:114331–114345, 2022.

[21] Trusted green charging project. `https://mobility.unternehmertum.de/en/\trusted-green-charging`. Accessed: 2023-2-14.

Part IV

Conclusions

"Plaudite, amici, comedia finita est"

Ludwig van Beethoven, said on this deathbed

9

Conclusions

This thesis has explored the integration of hardware security and Blockchain technology. The research has shown that hardware security is a critical component in ensuring the integrity and confidentiality of Blockchain systems. The use of HSMs, in particular TPM, can provide a secure foundation for Blockchain applications, protecting against attacks such as tampering, side-channel attacks, and reverse engineering.

The research has highlighted the advantages of using hardware security in Blockchain networks and the potential benefits of using hardware security in Blockchain systems are significant. The use of HSMs can enhance the security and privacy of Blockchain applications, enabling new use cases and applications that were previously not possible or that were possible to carry out by making the system adequately protected.

In addition, this thesis addresses the Blockchain Oracles through a secure sensor concept. Oracles are a critical component of Blockchain technology that enable the integration of external data into Blockchain applications. Indeed, they act as a bridge between the Blockchain and the outside world, providing a way for smart contracts to access and use data from external sources. This is important because Blockchain applications are typically isolated from the outside world, and cannot access data that is not stored on the Blockchain. One of the main challenges in the design of the Oracles is ensuring the accuracy and reliability of the data that is provided by them. This is particularly important in applications such as autonomous driving or energy trading, where even small errors or discrepancies in the data can have significant consequences.

This thesis has also approached new applications in the IIoT field. One of the most promising application of Blockchain is in energy trading field. In this kind of system, Blockchain has the ability to create a decentralized marketplace for energy transactions. This can enable peer-to-peer trading between energy producers and consumers, bypassing traditional intermediaries such as utilities and energy brokers. By eliminating these

intermediaries, energy trading can become more efficient and cost-effective, reducing transaction costs and increasing the profitability of energy producers. In addition, combining Blockchain and HSMs, new use cases in this field as tradable green certificates can be possible. Using the HSMs as shieled protection for these certificates, issuing them digitally signed, creates a hardware root of trust in this use case. Another benefit of using Blockchain in energy certificates trading is the ability to create smart contracts that automate the exchange process.

In short, the use of Blockchain in energy trading has the potential to transform the energy industry by providing a secure, transparent, and efficient platform for energy transactions. By enabling peer-to-peer trading and automating the trading process, Blockchain can reduce transaction costs, increase profitability, and promote environmental sustainability.

This thesis has also explored the use of Blockchain and HSMs in environments less considered by the literature but no less important, such as virtualised environments. In this sense, the use of this type of technologies in these architectures offers advantages in terms of security and integrity not considered until now. New challenges appear in this type of architectures, which have been addressed in this thesis by offering solutions based on mechanisms provided by HSMs.

During the course of this thesis, it has tried to shed light on new applications and use cases for Blockchain, in this sense systems such as the logistics of the wine transport chain, a system for the exchange of green energy certificates, or a system based on microservices on the geolocalization of electric vehicle charging stations have been proposed. These proposals have been analyzed from the perspective of the inclusion of two new factors so far little discussed in the Blockchain literature: the hardware security elements and their integration in this type of networks and the Blockchain Oracles and their critical role in this type of systems.

Overall, this thesis has contributed to the growing body of research on hardware security and Blockchain technology. As Blockchain technology continues to evolve and mature, the role of hardware security will become increasingly important in ensuring the security and privacy of these systems as well as the introduction of Oracles in the new Blockchain systems. In short, during this thesis, it has tried to close the gap that exists between the academic and scientific world and the industry, proposing new application use cases and providing key elements for the achievement of a greater maturity and implementation of this type of technologies in the IIoT field.

Bibliography

[1] Pereira, T., Barreto, L., Amaral, A. (2017). Network and information security challenges within Industry 4.0 paradigm. Procedia manufacturing, 13, 1253-1260.

[2] Muskaan, S. T., Singh, S. (2022, May). Challenges and Risks Associated with Public Key Infrastructure. In Edge Analytics: Select Proceedings of 26th International Conference—ADCOM 2020 (Vol. 869, p. 339). Springer Nature.

[3] Chen, Y., Lu, Y., Bulysheva, L., Kataev, M. Y. (2022). Applications of blockchain in industry 4.0: A review. Information Systems Frontiers, 1-15.

[4] Viriyasitavat, W., Da Xu, L., Niyato, D., Bi, Z., Hoonsopon, D. (2022). Applications of blockchain in business processes: A comprehensive review. IEEE Access.

[5] Alkhudary, R., Brusset, X., Fenies, P. (2020). Blockchain in general management and economics: A systematic literature review. European Business Review, 32(4), 765-783.

[6] Hasselgren, A., Kralevska, K., Gligoroski, D., Pedersen, S. A., Faxvaag, A. (2020). Blockchain in healthcare and health sciences—A scoping review. International Journal of Medical Informatics, 134, 104040.

[7] Yildizbasi, A. (2021). Blockchain and renewable energy: Integration challenges in circular economy era. Renewable Energy, 176, 183-197.

[8] Hu, W., Chang, C. H., Sengupta, A., Bhunia, S., Kastner, R., Li, H. (2020). An overview of hardware security and trust: Threats, countermeasures, and design tools. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 40(6), 1010-1038.