Doctoral Dissertation
Doctoral Program in Information and Communication Technologies

Enabling technologies for secure IoT-as-a-Service business model

Author:

Santiago Iván de Diego de Diego

March, 2025

Supervisors:
Prof. Gabriel Maciá-Fernández
PhD. Cristina Regueiro Senderos

Editor: Universidad de Granada. Tesis Doctorales
Autor: Santiago Iván de Diego de Diego
ISBN: 978-84-1195-910-0
URI: https://hdl.handle.net/10481/108591

This page was intentionally left blank

Declaration

I hereby declare that the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Santiago de Diego

2025

This page was intentionally left blank.

I would like to dedicate this thesis to my loving parents. Also, to my thesis directors Gabriel and Cristina for their support and continuous effort to improve this research work. Finally, to Ohiane for her love and support.

This page was intentionally left blank.

Acknowledgments

This work has been partially supported by:

- Izertis through the SSI4.0 project, which is a collaborative project co-funded by the Department of Economic Development, Sustainability and Environment of the Vice-Ministry of Technology, Innovation and Competitiveness of the Basque Government within the HAZITEK program. (File: ZE-2020/00020).
- The Government of the Basque Country under the ELKARTEK program, project TRUSTIND (KK-2020/00054) and by MINECO (Ministry of Economy and Competitiveness) through project TIN2017-83494-R.
- Research project PID2020-114495RB-I00 (SICRAC) funded by MCIN/ AEI /10.13039/501100011033 through the Spanish Government-Ministry of Science and Innovation
- Project AI4ES-2021(CER-20211030): Red de Excelencia en Tecnologías Habilitadoras basadas en el Dato. Partially supported by the Spanish Government-Ministry of Science and Innovation.

This page was intentionally left blank.

Abstract

The IoT-as-a-Service (IoTaaS) is an innovative business model that proposes to offer IoT devices on demand, with considerable cost savings and resource optimization, by enabling different applications to reuse existing devices. Despite the term having already been coined by both industry and academia, there is no formal analysis of the implications that this model has from a technological point of view.

Industry 4.0, also known as the Fourth Industrial Revolution, refers to the current phase of industrial transformation characterized by the integration of advanced digital technologies into manufacturing and industrial processes. It is set to modernize industrial processes as we know them today. This modernization goes hand in hand with the digitalization of industry and is closely related to the IoTaaS, the latter serving as an enabler and accelerator for Industry 4.0 initiatives. By providing accessible, scalable, and flexible IoT solutions, IoTaaS lowers down the entrance barriers for Industry 4.0 technologies and supports the digital transformation of manufacturing and industrial processes.

The implementation of the IoTaaS presents numerous technological challenges, with security standing out as a critical concern. Within the scope of security, identity management emerges as a fundamental issue. This issue extends to Industry 4.0 environments, where the digital identification of various devices integrated into the manufacturing process becomes crucial. The complexity of accurately and securely identifying and authenticating the myriad of interconnected devices poses a significant obstacle in both IoTaaS and Industry 4.0 implementations, stressing the need for robust identity management solutions.

This thesis makes two main contributions in clarifying this field that interconnects IoTaaS, Industry 4.0 and security. The first contribution is to formalize the technological implications of IoTaaS, identifying its technological challenges, describing them and

giving potential directions for the main problems. The second contribution is related to the problem of security in IoT environments, with a main focus on the identity management problem.

Here, the usage of Self-Sovereign Identity (SSI) schemes has been proposed to provide better privacy and scalability than traditional identity paradigms, which is especially important in the IoT owing to its characteristics. Verifiable credentials and decentralized identifiers, which are part of the SSI concept, allow decentralized identification and characterization of the devices (commonly IIoT devices) that make up Industry 4.0. However, some use cases in the Industry 4.0 cannot be modelled with standard SSI schemes. Despite the fact that delegated credentials have already been defined in the W3C standard for verifiable credentials, current technologies present some important limitations that make them non-implementable. This thesis analyses these limitations in the context of the problem of building delegated credentials for the Industry 4.0, and proposes an alternative based on an Hyperledger Aries RFC, bypassing these limitations.

Based on the previous problem of delegated credentials, a new problem arises. Current standard SSI protocols and procedures assume that individuals store only their own identity, failing to provide an accurate solution for the identity management of groups where participants might use credentials from different identities and collaborate to meet a set of verifier´s requirements. The identification of groups has been identified as another challenge for the IoT. Consequently, the present thesis also introduces the concept of Collaborative Credentials (CCs) to formalize identity management procedures that model the collaboration within a group of participants. CCs allow to leverage use cases requiring collaboration that cannot be solved with standard SSI verifiable credentials, increase the privacy of group participants, and enable the development of a software framework that any verifier/holder could use to generate a generic application.

To sum up, in this thesis we formally analyse the IoTaaS business model, identifying and detailing its main technological challenges. In addition, we tackle the identity problem of this business model and propose an SSI-based identity management system, which is compliant with the existing standards from the W3C. As part of the identity problem, delegation schemes and the use of CCs are also analysed. Finally, the identity model of the

IoTaaS is evaluated in terms of performance, as well as some tests have been conducted to study the feasibility of the use of credential delegation and CCs.

This page was intentionally left blank.

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

This introductory chapter sets the stage for this thesis by providing a comprehensive overview of its key elements. The chapter begins by presenting the context in which this research emerges, focusing on the intersection of Internet of Things (IoT), security, and as-a-service business models. Following this contextualization, the chapter delves into the specific problem addressed by the thesis, highlighting its significance and relevance in the current technological landscape. The chapter then proceeds to formalize the research objectives, outlining the primary goals and scope of the thesis. Subsequently, it describes the methodology employed to tackle the research questions, detailing the approaches and techniques utilized throughout the investigation. The chapter concludes by presenting the main results and contributions of the thesis, by emphasizing the publications that have stemmed from this research. Finally, the structure of the document is detailed.

1.1 The Internet of Things (IoT)

The expansion of the IoT is remarkable nowadays, with its rapid proliferation revolutionizing various sectors, including healthcare, manufacturing, and smart cities. In an era of ubiquitous connectivity, IoT has emerged as a transformative force, revolutionizing the way we interact with the digital world. The IoT has the potential to reshape our daily lives, revolutionize industries, and pave the way for a future where technology is integrated into every facet of our lives. From smart homes that optimize energy consumption and enhance security, to intelligent transportation systems that improve traffic management and reduce accidents, the IoT promises to enhance efficiency, productivity, and convenience across a wide spectrum of applications.

The IoT refers to a vast network of physical objects embedded with sensors, software, and network connectivity that enables these devices to collect and exchange data [1]. This

interconnected ecosystem includes a wide range of devices, from everyday household items, like smart thermostats and lighting systems, to sophisticated industrial tools and transportation system, which is commonly coined as Industrial Internet of Things (IIoT). At its core, IoT is characterized by the ability of devices to communicate with each other and with other internet-enabled systems without human intervention. This communication is facilitated through various technologies, including Wi-Fi, bluetooth, cellular networks, and other internet protocols.

To better understand IoT, it is crucial to distinguish it from related concepts. To name a few examples, traditional Internet connected devices, such as smartphones and computers that can access the Internet, are not considered IoT devices unless they are specifically designed to collect and exchange data with other devices autonomously. Secondly, standalone smart devices, which are, devices that are "smart" but do not communicate with other devices or systems, are not part of the IoT ecosystem either. Finally, simple data collection without analysis or action are not part of the IoT ecosystem, as IoT goes beyond mere data collection, involving the analysis and utilization of collected data to drive actions or insights.

The IoT is characterized by eight features [2], which are: *ubiquity, mobility, unattendance, constriction, interdependence, myriad, diversity, intimacy*. However, to put the reader in context, a few lines are written about these features:

1) **Ubiquity** means that IoT devices are now part of our lives without even noticing it, they are everywhere, and we use them and have become an indispensable tool for our daily basis.
2) **Mobility** means that many IoT devices are commonly used in a mobile environment, which usually implies that these devices hop from one environment to another.
3) IoT devices are also **unattended** as they have been thought to be operating during long periods of time without any human interaction.
4) As one of the most fundamental features, IoT devices are usually **constrained** in terms of computing and storage resources. Because of the mobility feature, they are usually forced to work in environments where there are no charging options, which implies that they are even more constrained in terms of saving resources.
5) Another very important feature to be considered is **interdependence**, which means that IoT devices are connected each other and the outputs from one of them could be the inputs for another.
6) **Myriad** feature is quite related to ubiquity, because it means that there are many IoT devices, and they manage a huge amount of data.

7) **Diversity** is synonym of heterogeneity, and it means that IoT devices are designed and made by different manufacturers, following different standards, and often using different protocols, and even having different uses.
8) **Intimacy** is related with the privacy risks for the user of an IoT device because this device stores private information about its user, such as blood pressure or heart rate in case of a healthcare device.

Despite its potential, the IoT faces several significant challenges [3], which are a consequence of its unique features described above. One of these challenges is interoperability. The lack of unified standards among different manufacturers and protocols creates an integration burden. This fragmentation can lead to inefficiencies and increased complexity in IoT deployments. IoT ecosystems often involve complex interactions between multiple devices, platforms, and protocols, complicating system design, deployment, and maintenance. Connectivity is another challenge, as maintaining reliable and consistent connectivity across diverse environments and geographies may present significant issues. This includes addressing issues related to cellular coverage, network reliability, and power consumption. Another challenge is scalability. As the number of connected devices grows exponentially, IoT systems must be able to handle this massive scale efficiently. This includes managing network capacity, data processing, and storage requirements. Scalability challenges also extend to device management, activation, monitoring, and maintenance. Scaling IoT solutions can also be expensive, particularly when considering the need for robust security measures, data storage, and network infrastructure. Scalability and sustainability are closely intertwined in the context of the IoT, with each concept significantly impacting the other. As the scale of IoT deployments increases, the environmental impact associated with these technologies can become substantial. This impact spans various stages, starting with the energy-intensive operations of data centers that process and store vast amounts of data generated by IoT devices. Additionally, the production of numerous IoT devices involves the extraction and use of raw materials, manufacturing processes, and transportation, all of which contribute to environmental degradation. Furthermore, the disposal of these devices at the end of their lifecycle poses significant challenges, as many contain hazardous materials that can harm ecosystems if not properly managed. Therefore, the growth of IoT necessitates careful consideration of sustainable practices to mitigate its environmental footprint. Consequently, scalable IoT implementations must prioritize sustainability to be viable in the long term. Another challenge is security. The proliferation of IoT devices creates an expanded attack surface for cybercriminals. Security challenges include device vulnerabilities due to limited processing power and memory, weak encryption and insufficient security protocols, physical security risks, as devices are often deployed in unguarded locations, long-running sessions that increase the likelihood of attacks, privacy concerns, amongst other.

This thesis aims to address two critical challenges: sustainability and security. On the one side, to understand the magnitude of the sustainability issue, remember that IoT is being integrated into numerous solutions owing to its ubiquity. According to the study in [4], the number of connected devices around the world is expected to increase from 19.5 billion in 2025 to 40.1 billion in 2030, with a Compound Annual Growth Rate (CAGR) of 16%. Considering the expected number of IoT devices, it is likely that many of them remain underused. It is therefore necessary to think about ideas and solutions to reuse existing devices, rather than creating new ones, thus avoiding the negative consequences of overpopulating our environment with IoT devices [5] and fostering cost reductions as well. To tackle this issue, one promising approach involves implementing mechanisms for repurposing existing devices or adopting an "as-a-service" business model for current hardware. This strategy would optimize device utilization, effectively reducing the overall number of devices required by maximizing the efficiency of existing resources. For example, an IoT device measuring temperature at a specific location may be useful for researchers interested in its data for their own applications. Being able to reuse existing devices has a positive impact on the sustainability by encouraging the four R's [6]: Reusing, Repairing, Refurbishing and Recycling.

In parallel with the increased popularity and ubiquity of the IoT, many issues and challenges that need to be carefully addressed have arisen regarding security. By 2030, IoT security services will be a lucrative US$41.2 billion revenue market, with a CAGR of 25% from 2025 [4]. As billions of devices, ranging from smartphones and wearables to household appliances and industrial sensors, integrate into our daily lives, the need for security in this context is becoming increasingly important. Within security, the focus is put into identity management, because it is fundamental to achieve a certain security level. Identity management is crucial for scalability because it ensures that as a system grows, it can efficiently manage and authenticate a larger number of users without compromising security. By focusing on this aspect, the research seeks to enhance both the scalability and security of the system while minimizing unnecessary hardware proliferation. The following sections within the present chapter will shed more light to these two challenges.

1.1.1 Security in the IoT

As stated by numerous authors [2][7], it is not trivial to achieve an overall security level in IoT ecosystems. Some IoT devices, such as wearables, keep sensitive information from their users and, for this reason, attackers are increasingly targeting these devices. Unfortunately, some well-known security incidents such as Stuxnet [8], the Mirai botnet, and other IoT zombie-related attacks [9] that have infected millions of IoT devices emphasize the necessity of a security point of view when developing IoT-related solutions. Other more generalist attacks include data theft, device malfunctioning, and remote control [9]. In this sense, solutions that tackle security problems associated with IoT ecosystems are welcomed.

The security model based on IoT features [2] gives us an important overview on how these devices can be hacked and what kind of measures can be adopted to protect them. These features characterise IoT devices, and it is important to take them into account when deploying security measures. The security implications of these features are described below:

1) **Ubiquity**: this pervasive presence means that any security breach can have widespread and immediate impacts. Traditional security measures, designed for more centralized and noticeable systems, may not be effective in such a dispersed and integrated environment.
2) **Mobility**: unlike stationary systems, these devices must adapt to diverse security protocols and potential threats dynamically. This mobility requires security solutions that are flexible and capable of real-time adaptation, which is not a common requirement in traditional computing.
3) **Unattendance:** this autonomy increases the risk of physical tampering and necessitates robust, tamper-resistant security measures. Traditional systems, which typically benefit from regular human oversight, do not face this level of risk and thus do not require such stringent physical security protocols.
4) **Constraint**: they often operate in environments without reliable power sources, further limiting their capacity to implement resource-intensive security measures. Traditional security solutions, which assume extensive computational resources, are often unsuitable for these constrained environments. Therefore, IoT security solutions must be lightweight and efficient, balancing security with resource consumption.
5) **Interdependence**: this feature has enabled what is commonly known as an *interdependence attack*, whereby altering the state of one device, the attacker is able to alter the state of other devices interdependent of the first one. Authors also provide detailed examples about how these kinds of attacks work.
6) **Myriad**: protecting IoT data and ensuring the integrity of numerous devices requires scalable and robust security solutions. Traditional systems, which typically manage fewer devices and less data, do not require the same level of scalability and robustness.
7) **Diversity**: this heterogeneity complicates the implementation of uniform security measures. Traditional systems, which often benefit from standardized protocols and more homogeneous environments, do not face this level of complexity. IoT security solutions must be adaptable to a wide range of devices and standards.
8) **Intimacy**: with the sharp increase of wearables in the last years and the raising of concerns about user privacy, intimacy has become one of the top-priority features to consider when designing an IoT solution. This intimacy raises significant privacy concerns, making data protection a top priority. Traditional systems, which may not handle such personal data, do not require the same

level of privacy protection. IoT solutions must prioritize user privacy and data security to a greater extent.

These features are extremely important because every use case involving IoT devices, in particular the study of the present work, must consider these features to avoid security breaches. Without robust security measures, ecosystems are vulnerable to cyber threats, data breaches, and malicious attacks, which can lead to significant financial losses, reputational damage, and loss of user trust. Therefore, it is crucial to prioritize secure IoT scenarios to ensure the integrity, confidentiality, and availability of data and services and it is paramount to consider scenarios with secure IoT, instead of only IoT.

1.1.2 Identity for the IoT

Identity management is one of the relevant problems that falls under the umbrella of security. It can be informally defined as the process of assigning identities to the different actors in an IT ecosystem and managing them in a secure way. The basic notion of digital identity serves the primary function of authenticating an entity's identity to enable the flow of information while verifying permissions, data integrity, and validity.

The identity should include information for the identification of the actor itself but also the context or the own features of the subject. As an example, brought from the IoT scenario, when considering a smart temperature sensor, it is not only essential to know its ID for authorization (e.g., writing to a database) but it is also relevant to understand its specific characteristics (such as measurement ranges, maximum supported temperature, compliance with regulations, etc.) and its current state (e.g., any malfunctions or performance losses).

In the context of digital identity, new solutions have begun to emerge, primarily focused on personal identity but with potential applications in the industrial domain. Currently, personal identity data is predominantly stored in centralized databases where users voluntarily expose their information. However, there is a growing need for a model that restores control to users over their own information and identity, dismantling the information silos that can be very attractive for being exploited by malicious actors.

Identity is a relevant aspect in the security process, as many attacks that are based on techniques such as impersonation and identity theft can be prevented by a solid identity management. For example, the Sybil attack also affects the IoT [11] and is directly caused by deficient identity management. Good identity management practices also help to guarantee non-repudiation by identifying the different actors in the system. To provide some insights about the challenges associated with identity, it is estimated that the cost of identity verification processes is expected to be $0.20 per verification globally in 2025 [12], without taking into account the costs associated with storage, protection, breaches, and

regulations. By adding IoT into the equation, the numbers are completely exorbitant. Hence, it is necessary to manage the identity of all existing devices.

The problem of identity for IoT devices is multifaceted due to the sheer number of devices and their inability to manage their own identities like humans do. Unlike people, a unique approach that is valid for the wide range of contexts in which IoT operates is needed, because IoT lacks simple adaptability or intelligence. This context-specific identity is crucial for ensuring that each device can be accurately identified and authenticated within its specific environment. Without a robust system to manage these identities, the risk of security breaches and operational inefficiencies increases, making it a critical challenge in the IoT ecosystem. Consequently, any technological proposal made in the field of IoT must give special consideration to the problem of identity.

Furthermore, identity solutions are primarily focused on managing individual identity. In the context of IoT, where collaboration between devices is necessary to carry out complex operations efficiently, it is essential to develop identity mechanisms for groups of devices that also allow for collaboration. Consequently, the concept of Collaborative Credentials (CCs) (introduced in this thesis) emerges as a crucial solution to address the complex identity management challenges on IoT, particularly for managing group identities. As the IoT ecosystem continues to expand, identity management systems struggle to efficiently handle the identification and authentication of device groups. This gap in the literature is significant, as it will be studied in Chapter 2, as group identification has been identified as a necessity for the IoT.

1.2 As-a-Service business model

As introduced before, due to the exponential growth in the number of IoT devices and the importance of sustainability, there is an increasing necessity on reusing and optimizing the usage of existing IoT devices. The “as-a-service” business model emerges as a possible solution to this problem. The "as-a-service" business model refers to a service-oriented approach in which a company provides a specific service or product to customers through a subscription or usage-based model, typically delivered over the Internet or a network. Overall, the "as-a-service" business model allows companies to deliver services or products efficiently, focusing on core competencies while providing customers with flexible access to the service without incurring on infrastructure and management costs. It moves away from traditional ownership-based models and focuses on providing services on a non-ownership basis.

The “as-a-service” label has become incredibly popular nowadays [13]. In the context of the Internet, it is not necessary to own a resource to use it. In “as-a-service” models, users of resources can reduce costs, as they avoid investing on its acquisition, while owners of resources can monetize their investments by offering services based on those assets to

interested users. In other words, "as-a-service" often implies a win-win situation for both users and owners of resources.

Some examples of successful “as-a-service” models [14] are:

- Software-as-a-Service (SaaS): cloud-based software applications accessible via subscription.
- Platform-as-a-Service (PaaS): platforms for developing, deploying, and managing applications.
- Infrastructure-as-a-Service (IaaS): cloud-based infrastructure resources (e.g., virtual servers, storage) provided on-demand.

The "as-a-service" business model usually involves the following key elements:

- Service/product provision: the service provider offers a particular service or product. The specific nature of the service or product depends on the industry and market needs.
- Subscription/usage-based pricing: instead of selling the service or product, the service provider charges customers based on a subscription model or usage metrics. Subscriptions can be daily, weekly, monthly, annually, etc., and usage-based models involve billing according to the resources consumed, such as storage, processing power, or bandwidth.
- On-demand access: the service or product is accessible to customers whenever needed, typically over the Internet or a network. This allows customers to utilize the service without requiring substantial investment or ownership. The service provider maintains the infrastructure and resources necessary to deliver the service.
- Scalability and flexibility: the "as-a-service" model offers scalability and flexibility, enabling customers to adjust their service usage based on demand. They can easily scale up or down their consumption as their needs change, avoiding the constraints of fixed resources.
- Maintenance and updates: the service provider is responsible for maintaining and updating the service or product. This includes regular updates, bug fixes, security patches, and enhancements, ensuring that customers have access to the latest version and features without additional effort on their part.
- Service-Level Agreements (SLAs): to establish clear expectations, the service provider typically defines SLAs. SLAs outline the quality, availability, and performance metrics of the service, specifying the level of support, uptime, response time, and other key parameters agreed upon with customers.

The Netflix use-case [15] is a perfect example of a successful “as-a-service” solution present in the market, where it is shown that a traditional use-case (watching films on the Internet) can be redefined as a subscription-based business model. However, other business

models could also be mentioned, such as the pay-per-use model, where users pay for the time they use the service. Another widely known "as-a-service" solution is Amazon Web Services (AWS) [16], where any user can use cloud computing and storage capabilities, thus paying only for the utilized amount of resources or traffic.

In this context, the IoT-as-a-Service (IoTaaS) business model emerges as another "as-a-service" business model, adding special value when it comes to enabling different applications aimed at optimizing the use of IoT devices. Thus, it intends to fill the existing gap regarding IoT reutilization and sustainability by proposing a business model which value proposal is, amongst other, to monetize and reuse existing devices. Revenue from IoT data and analytics services will grow at a rate of 19% per year, increasing from US$91.9 billion in 2025 to almost US$218 billion by the end of the decade [4]. The IoTaaS intends to offer devices on demand to a group of consumers. Despite the research community have scratched the surface over this term [17] [18], these references do not go on a deep analysis into the business model description, the technological challenges and its operation. Therefore, the IoTaaS business model is analysed in this thesis, providing not only a formal model, but also discussing key enabling technologies for its implementation.

1.3 Problem statement and objectives

As explained in the present chapter, the IoT has emerged as a pivotal technological landscape that not only presents numerous opportunities but also challenges that need to be fully addressed. Among these challenges, sustainability and security stand out as critical areas requiring comprehensive solutions. However, the IoTaaS model also faces its own set of hurdles that demand careful consideration and precises of enabling technologies that need to be properly analyzed. IoT security is a critical issue that must be addressed to ensure the success of IoTaaS. The unique characteristics of IoT environments make standard security solutions from other scenarios inadequate or ineffective. One of the cornerstone issues in IoT security is identity management for devices. This challenge extends beyond individual device authentication to encompass the management of device groups, an area that remains largely unexplored.

The following research questions (RQ) have been identified as relevant in this context:

RQ1: Can the IoTaaS be generalized and formalized?

RQ2: What are the use cases that can benefit from the IoTaaS formalization?

RQ3: What are the technological challenges associated to the IoTaaS?

RQ4: Which of the identified challenges in RQ3 require a novel approach, not solved in the literature?

RQ5: Which technological challenges are fundamental to achieve security for the IoTaaS?

RQ6: Can we solve the challenges obtained in RQ4 and RQ5?

The **primary aim** of this thesis is:

"To take a deep dive into the IoT-as-a-Service (IoTaaS) business model from a technical perspective by identifying its associated technological challenges, proposing solutions for some of these challenges from a security perspective and evaluating these solutions."

This main objective is divided into these five secondary objectives:

A) IoT-as-a-Service model formalization. This objective focuses on the formalization of the IoTaaS model, identifying and defining the key actors involved. The goal is to create a comprehensive framework that outlines the roles, responsibilities, and interactions of various stakeholders, including service providers and consumers. This formalization would serve as the foundation for understanding the operational dynamics and governance of IoTaaS.

B) IoT-as-a-Service model analysis. This objective focuses on the analysis of the model, the identification and the definition of technological challenges and subsequent solutions for the IoT-as-a-Service. The goal is to provide innovative solutions for some challenges contributing an advancement to the current state of the art.

C) Identity Model for IoT-as-a-Service. Recognizing the paramount importance of identity management in the security domain, this objective aims to develop a robust identity model tailored for IoTaaS. Effective identity management is crucial for ensuring secure access and communication within IoT ecosystems. This objective will address the challenges associated with identity verification, authentication, and authorization, providing a secure solution that enhances trust and reliability in IoTaaS deployments.

D) Solution for the identity problem for groups within IoT-as-a-Service. Building on the identity model, this objective seeks to solve the identity management issues specific to groups of devices that may create a group and collaborate, within the IoTaaS business model, instead of only individual devices. Group identity management, which imply managing collective identities and require real time collaboration, presents unique challenges.

E) Performance evaluation of the proposals. Conduct a thorough performance evaluation of the proposed identity management solutions for the IoTaaS, in order to prove their feasibility in a real environment.

1.4 Methodology

The methodology used in this thesis is as follows. First, a comprehensive study of the state of the art has been done, which has helped us to identify the deficiencies in the current definition of the IoTaaS business model, as well as its associated technological challenges. Consequently, we have identified some challenges to be solved. Within the identified challenges, we have given a special focus to the problem of identity, mainly because it is fundamental for achieving security in any system and, as discussed in Chapter 2, it is unsolved for the IoTaaS. Consequently, an identity model for the IoTaaS has been proposed. Furthermore, we have identified some aspects, such as the concept of Collaborative Credentials, that are useful for the IoTaaS and are not fully covered with standard decentralized identity models and proposed an architectural and implementation solution, as well as use cases that can be derived from them. The proposed solutions have been studied both from an architectural and implementation view, by providing a minimum set of proofs of concepts to evaluate the feasibility of such implementations. Table 1-1 shows the planning of the works in the thesis with the details of the different actions taken. Following Figure 1-1 shows the GANTT diagram with the different steps of the thesis.

Table 1-1: *Planning of the works in the Thesis*

Year	***Actions***
2019/2020	• Definition of research objectives within the IoTaaS (A1) • Analysis of the state of the art in the selected topic (A2)
2020/2021	• Article sent to a Q1 scientific journal, IEEE Access, about the IoTaaS business model formalization, its associated challenges, and a proposal for an identity management system and another article sent to JNIC (Jornadas Nacionales de Investigación en Ciberseguridad) (A3) • Seeking significant contributions in their line of research (A4)
2021/2022	• Article sent to 19th International Conference on Security and Cryptography (SECRYPT) conference, about the current implementation issues and gaps of verifiable credentials delegation, discovered as a result of A4 (A5) • Definition of the concept of “Collaborative Credentials”, extending the concept of delegation (A6) • Application of “Collaborative Credentials” to the Industry 4.0 (A7)

2022/2023	• Research about "Collaborative Credentials, or CCs" (A8) • "Collaborative Credentials" within the IoTaaS (A9) • Article sent to a Q1 scientific journal, Computer Networks, tackling the concept of CCs and its application to the IoT (A10)
2023/2024	• Analyse other use cases for the IoTaaS and the secure identity solutions investigated in previous courses (A11) • Article sent to 6th International Congress on Blockchain and Applications (Blockchain24), with some performance evaluations of a SSI solution in an Arduino device simulating an IoT (A12) • Write thesis report (A13)
2024/2025	• Write thesis report (A13)

1.5 Publications and funding

This research work has received support from different projects, which have specific research objectives, which are:

i) SSI4.0 (ZE-2020/00020): Sovereign Digital Identity and Data Sovereignty in Industry 4.0. Partially supported by the Basque Country Government under the HAZITEK program.
ii) TRUSTIND (KK-2020/00054): Creating Trust in the Industrial Digital Transformation. Partially supported by the Basque Country Government under the ELKARTEK program.
iii) SICRAC (PID2020-114495RB-I00): Collaborative Credentials in Self-Sovereign Identity systems for IoT access control. Funded by the Spanish Government-Ministry of Science and Innovation.
iv) AI4ES-2021(CER-20211030): Red de Excelencia en Tecnologías Habilitadoras basadas en el Dato. Partially supported by the Spanish Government-Ministry of Science and Innovation.

The following research articles have been published within the scope of the present thesis and correspond to the following activities presented in Section 1.4:

- A3: De Diego, S., Regueiro, C., & Maciá-Fernández, G. (2021). Enabling identity for the IoT-as-a-service business model. *IEEE Access, vol. 9, pp. 159965-159975, 2021, doi: 10.1109/ACCESS.2021.3131012*.
- A3: De Diego, S., Regueiro, C., & Maciá-Fernández, G. (2022). IoT Como Servicio. In Jornadas Nacionales de Investigación en Ciberseguridad (JNIC).

- A5: de Diego, S., Lage, O., Regueiro, C., Anguita, S., & Maciá-Fernández, G. (2022). Bypassing Current Limitations for Implementing a Credential Delegation for the Industry 4.0. In *SECRYPT* (pp. 485-490).
- A10: de Diego, S., Regueiro, C., & Maciá-Fernández, G. (2024). Collaborative Credentials for the Internet of Things. Computer Networks, 110629.
- A12: de Diego, S., Regueiro, C., Maciá-Fernández, G. (2025). An Authentication System Based on Self-sovereign Identity for Vehicle-to-Vehicle (V2V) Communications. In: Prieto, J., Vargas, R.P., Lage, O., Machado, J.M., Bálint, M. (eds) Blockchain and Applications, 6th International Congress. BLOCKCHAIN 2024. Lecture Notes in Networks and Systems, vol 1256. Springer, Cham. https://doi.org/10.1007/978-3-031-81928-5_2.

1.6 Structure of the document

The thesis is structured as follows: Chapter 1 has justified the need for the study presented in this thesis, presented the objectives and methodology, and disclosed the publications and funding. Chapter 2 presents the state of the art and some introductory concepts. Chapter 30 delves into the IoTaaS business model formalization and analysis. Chapter 4 properly justify the use of Self-Sovereign Identity (SSI) to solve the identity problem within the IoTaaS and proposes a protocol based on SSI for this business model. Chapter 5 deeps into the concept of Collaborative Credentials, which are a special type of verifiable credentials used in collaborative scenarios that require collaboration among different actors and shows some interesting use cases where this concept can be applied. It also defines a formal model for Collaborative Credentials, including their structure and lifecycle. Finally, it proposes a simple but functional implementation for Collaborative Credentials. Chapter 6 shows the performance evaluation, where some implementation tests are conducted to prove the feasibility of the designs proposed in previous chapters. Finally, conclusions, acronyms and references are provided.

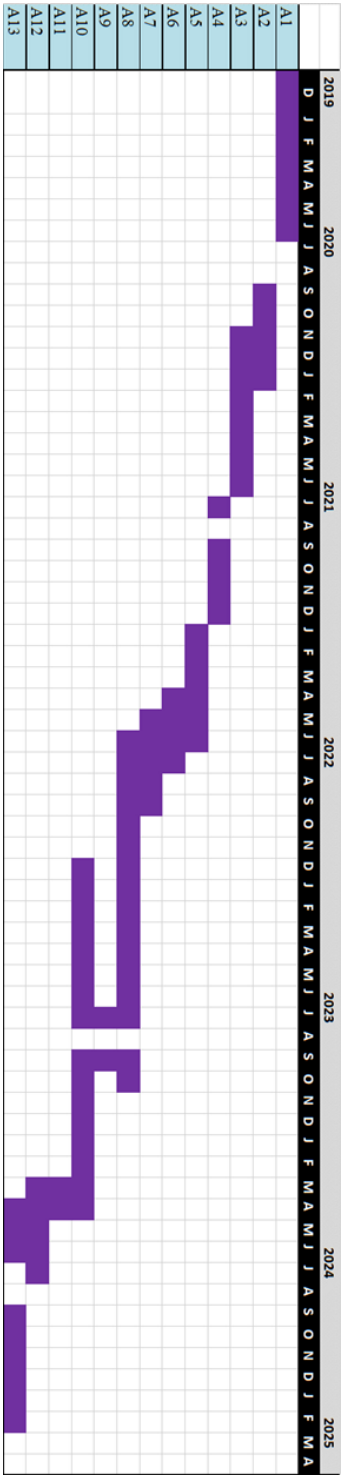

Figure 1-1: *GANTT diagram for the activities of the thesis*

Chapter 2.

Basic concepts and state of the art

In this chapter, we explore the theoretical keystones and recent advancements relevant to our research topic. The state of the art review will be structured to provide a logical progression through the key concepts and developments in the field of study. First, some basic concepts are introduced, helping the reader to understand the basic building blocks that form this thesis, i.e.: the Self-Sovereign Identity (SSI) and Distributed Ledger technologies and its application to SSI. Then, we will narrow our focus to more recent and directly relevant research about the IoTaaS business model, SSI and IoT and the concept of Collaborative Credentials, with the main goal of articulating the specific gaps or problems that our thesis aims to address.

2.1 Basic concepts

This section explores some basic concepts that are necessary to understand the thesis. By establishing a clear and comprehensive understanding of these fundamental ideas, we aim to provide the reader with the necessary context and background to fully understand the research findings presented in later chapters.

2.1.1 Self-Sovereign Identity

SSI arose from the aspiration for self-determination and self-governance. As our lives become increasingly intertwined with digital systems, the need for individuals to have control over their digital identities has become a priority. SSI challenges the traditional model where third parties, such as governments or corporations, have primary control over personal data. The concept of SSI is rooted in the belief that individuals have the right to an identity independent of reliance on a third-party identity provider. This philosophy

aligns with idealist ideas, pushing back against the notion that our identities are owned or controlled by the state or other entities. The SSI is based on 10 principles to ensure the user control over his own identity [19], and these can be applied to people or even machines, such IoT devices:

1. Existence: Users must have an independent existence. SSI is based on the identity as it only refers to attributes of the identity that already exist. In the context of IoT, the identity refers only to IoT devices from the point of manufacture, allowing them to exist.
2. Control: Users must control their identities as they are the ultimate authorities on their identities. They should always be able to refer, update or even hide it. IoT devices, which increasingly include higher levels of intelligence, should control their own attributes, increasing security and confidentiality.
3. Access: Users must have access to their own attributes, with no hidden data and no gatekeepers. IoT devices should also be able to access all their own attributes when desired.
4. Transparency: Systems and algorithms must be transparent; anyone should be able to examine how they work.
5. Persistence: Identities must be long-lived. Nowadays, identities should last until they are outdated. In addition, user's identity should be modified or removed as appropriate over time. This fact is important for IoT devices as many of their features may be degraded over time and should no longer be part of the IoT identity.
6. Portability: Any identity that manages information and services shall be able to use the mechanisms that best suit its needs without losing any capability provided by the identity. The IoT devices shall be allowed to change the application that manages the identity at any time, and the identity information and purposes must remain the same.
7. Interoperability: Identities should be as widely usable as possible. Ideally, they should cross international boundaries creating global identities, without losing user control. As in the previous case, this is essential for IoT, as devices are manufactured in an organisation but will be used in a different one; their identity should be persistent among different organisations, different countries, etc.
8. Consent: Users must agree to the use of their identity; there should be a previous "consent".
9. Minimisation: Disclosure of claims must be minimised. Only required attributes should be shared, increasing confidentiality as best as possible.
10. Protection: The rights of users must be protected.

The political drive for SSI stems from growing concerns about data privacy, security, and individual rights in the digital age. Governments and policymakers are recognizing the

need for more user-centric approaches to digital identity management. The European Union has taken significant steps towards implementing SSI in its digital identity framework. In June 2021, the European Commission announced plans for Digital Identity Wallets as part of the revised eIDAS regulation [20][21]. This initiative aims to provide every EU citizen with a digital wallet that embodies many SSI principles. In addition, the European Blockchain Partnership, comprising 29 countries and the European Commission, is working on the European Blockchain Services Infrastructure (EBSI) to accelerate the creation of trustworthy cross-border digital services, which includes a framework for SSI, known as ESSIF (European Self-Sovereign Identity Framework) [22].

Decentralized Identifiers (DIDs) provide a standard way to create permanent, globally unique, crypto-verifiable identifiers for people and organizations under their own control. Consequently, DIDs are the first verifiable identifiers that do not require any registration authority. DIDs follow the format: *"did:" method-name ":" method-specific-id*, specified by the W3C standard for DIDs [23]. The *method-name* defines the kind of DID and can be of different types. For example, the *web* method allows organizations and individuals to use their domain names as the basis for DIDs, making it a relatively simple and accessible method for those already operating websites. As another example, organizations can create their own methods, like for example EBSI, which has created its own DID method: *EBSI*. The *key* method is a lightweight, self-contained DID format that does not require any external systems for resolution or verification. It is particularly useful for simple, one-time interactions or in situations with limited network access. The contained information within the *key* method allows the direct derivation of the value of the subject's public key that it identifies. The DID document includes the DID itself, the public key for the DID, any other public credentials the identity owner wishes to reveal, network interaction addresses (endpoints), authentication protocols, and the digital signature. The identity owner maintains control over the DID document by managing the associated private key. Unlike domain names, IP addresses, or phone numbers, no service provider rents a DID, and no one can take it away from the person who owns or controls the associated private key.

A Verifiable Data Registry (VDR) is a crucial component in the SSI ecosystem which serves as a secure and transparent system for managing data related to digital identities and VCs. The contents of a VDR are typically public, allowing anyone to access the necessary information for verification and they should be designed to be tamper-evident, ensuring the integrity of the stored data. Normally, the VDR contains information such as: information about credential issuers, public keys of trusted issuers, credential schemas, revocation registries and details of available credential types. A DID can be also looked up in VDR to retrieve a DID document, which describes the subject being identified. Private DIDs are also possible when involved parties want to keep their relationships private, so DIDs do not necessarily need to be recorded in the VDR.

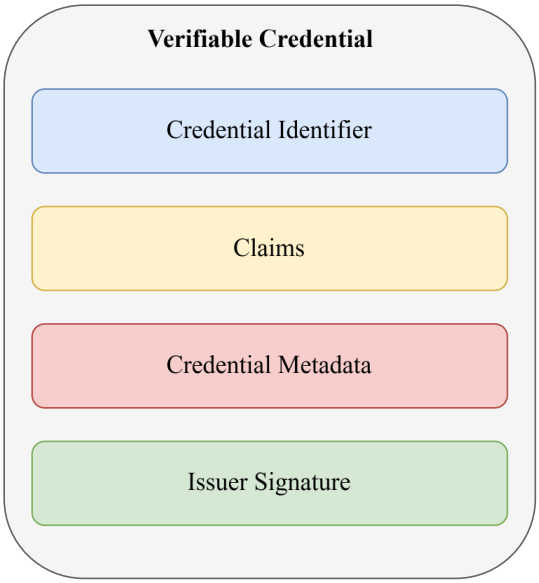

Figure 2-1: *Elements that compose a VC according to the W3C*

Many implementations of VDRs, blockchain-based VDRs, use decentralized technologies to enhance security and reduce reliance on central authorities. A blockchain-based VDR offers advantages to all actors involved in identity creation. Blockchain guarantees the integrity of the public information stored in the VDR, thanks to the by-design immutability. Standardized, shared registries simplify identity attribute verification, improving verification processes and reducing costs. However, practical challenges related to managing and safeguarding cryptographic keys persist. In cases of purely self-managed identity, if the user mishandles cryptographic seeds, they may need to reconstruct their identity from scratch, potentially losing data or attributes. This management complexity currently poses usability issues for non-technical users. To help the reader to understand the advantages of blockchain systems, some basic notions about Distributed Ledger Technologies (DLTs), like blockchain, are presented in Section 2.1.2.

The SSI works with the concept of *verifiable credentials* (VCs), which are the digital equivalent to traditional paper credentials, and are also verifiable. Figure 2-1 shows the different elements that compose a VC, according to the W3C [24]. The use of claims avoids sharing personal and undesired information with third parties, but only the required information. This is normally achieved by using zero-knowledge proof (ZKP) cryptography to prove that it has a specific attribute without having to show the attribute itself. The W3C [24] standard for VCs proposes at least three proof mechanisms: JSON web tokens (JWT) [25] secured using JSON web signatures (JWS), link data signatures [26] and Camenisch-Lysyanskaya zero-knowledge proofs [27]. More information about the relationship between VCs and JWT tokens can be read in Section 6.3.1 of the W3C standard. Technically, by using ZKPs, someone can use cryptography to generate a

Verifiable Presentation (VP) based on a VC without disclosing the attributes of the VC. Deeper information about zero-knowledge proofs and their relationship with SSI can be found in other works [28]. Figure 2-2, extracted from the W3C, is a JWT representing a VC. First of all, the header contains information about the signature algorithm used. Then, the payload contains the content of the VC, such as the different attributes of the VC, among other useful information.

```
// JWT Header
{
    "alg": "RS256",
    "typ": "JWT"
}
// JWT Payload
{
    "iss": "https://dmv.example.gov",
    "iat": 1262304000,
    "exp": 1483228800,
    "aud": "www.example.com",
    "sub": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "verifiableCredential": {
        "@context": "https://w3id.org/security/v1",
        "id": "http://example.gov/credentials/3732",
        "type":["VerifiableCredential", "ProofOfAgeCredential"],
        "issuer": "https://dmv.example.gov",
        "issuanceDate": "2010-01-01",
        "claim": {
           "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
           "ageOver": 21
     }
}
```

Figure 2-2: *JWT representing a VC, extracted from the W3C*

At the same time, a Verifiable Presentation (VP), according to the W3C, is composed by a set of elements, as shown in Figure 2-3, which are: i) Presentation identifier: is a unique identifier that helps to distinguish one VP from another, ensuring that each VP can be uniquely referenced and verified; ii) Verifiable Credentials: set of VCs that the *holder* desires to present; iii) Counter Signature: cryptographic proof provided by the *holder* of the VP that ensures the integrity and authenticity of the VP. A VP is a way to share VCs in a secure and privacy-preserving manner. It allows to prove the authenticity of the information without revealing more than necessary. Following, the roles that participate in an SSI ecosystem are presented.

First of all, the *issuer* is a trusted entity responsible for creating and distributing VCs. *Issuers* can be institutions such as governments, universities, banks, or healthcare providers. They validate information and securely issue VCs that attest to specific attributes or qualifications of an individual or organization. For example, a university might issue a digital diploma, or a government could issue a digital passport.

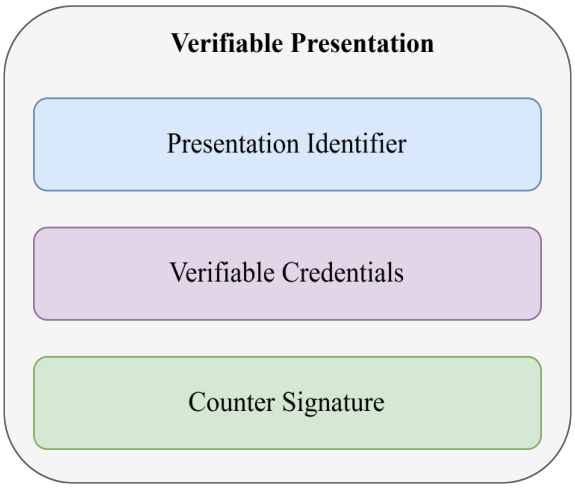

Figure 2-3: *Elements that compose a VP according to the W3C*

Secondly, the *holder*, who is typically an individual or organization that receives and possesses VC, has complete control over their digital identity and personal data, storing their credentials in digital wallets on their devices A wallet is a private user storage that contains VCs and identity-related information, as well as the *holder´*s private keys. Each corresponding public key is associated with an address (DID). The *holder* role empowers users to manage their own identity information, deciding when and with whom to share specific pieces of data. *Holders* can aggregate credentials from various *issuers*, building a comprehensive digital identity under their own control. The *holder´*s ability to selectively disclose information as needed is a key feature of SSI, enhancing privacy and user autonomy. A *claim* proves certain attributes of the *holder* without disclosing the entire credential information.

The third role in the SSI ecosystem is the *verifier*. *Verifiers* are entities that request and verify VCs presented by *holders* in the form of a VP. They can be organizations or individuals acting in professional capacities, such as employers verifying job applicants' qualifications or businesses confirming customers' age for restricted services. *Verifiers* use the VPs to make informed decisions, and they can easily confirm the authenticity and validity of these VPs by interacting directly with the *issuer*. This direct verification process eliminates the need for manual checks and intermediaries, streamlining identity verification while maintaining security and privacy. Finally, Figure 2-4 draws the different actors that characterize the SSI according to the W3C:

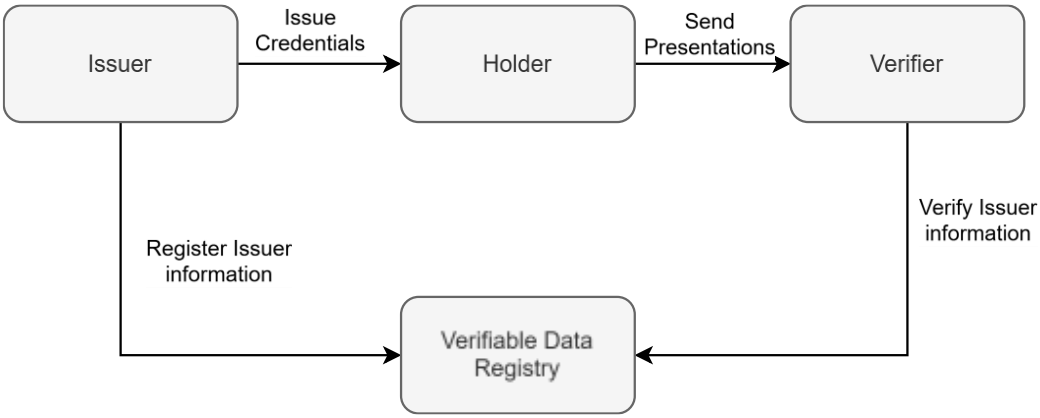

Figure 2-4: *Roles that characterize the SSI according to the W3C*

Therefore, to sum up, VCs are issued to a *holder* by a trustable *issuer* as a digital equivalent to traditional paper credentials. A *verifier* could use a VDR to check whether the public keys to be used to verify the signatures in the *VCs* presented by a *holder* are valid.

2.1.2 Distributed Ledger Technologies (DLTs)

As explained, a VDR implemented as a Distributed Ledger Technology (DLT), particularly a blockchain, has advantages in terms of security. Therefore, the present section draws some preliminaries about DLTs. DLTs can be defined as databases for recording information that is not controlled by a single entity. Unlike traditional centralized databases, DLTs allows data to be both decentralized (stored in multiple locations) and distributed (connected and able to communicate) either privately or publicly. They usually rely on peer-to-peer networks and consensus algorithms to ensure data replication across nodes.

Although a DLT and a blockchain are not the same concept, they are similar. Blockchains are included within the definition of DLTs, so a blockchain is a type of DLT. Both are decentralized and digitalized ledgers. However, they differ in specific ways. DLT is a multi-participant database without a central authority, enhancing transparency and security. Blockchain, on the other hand, is a specific type of DLT where data is organized into blocks, each one being cryptographically linked to the previous one, thus preserving integrity.

Blockchain is a technology conceived by Satoshi Nakamoto in 2008 and presented as a set of algorithms and architectures that allowed the creation of a decentralized cryptocurrency based on cryptographic techniques called Bitcoin [29]. In Bitcoin, anyone who wants to add new blocks to the chain must solve a difficult puzzle that requires significant computational power. Solving the puzzle "proves" that the "work" has been done using computer resources (Proof of Work or PoW). This initial consensus mechanism has evolved with numerous alternative proposals, such as Proof of Stake (PoS), or Practical Byzantine Fault Tolerance (PBF), which propose, among other advancements, to improve the mining time of PoW. Several authors [30][31] have surveyed the most important consensus protocols. This process is also known as mining. Mining is often a brute-force trial-and-error process, but successfully adding a block is rewarded in BTC. The new blocks are broadcasted to network nodes, checked, and verified, thus updating the blockchain's state for everyone. Nakamoto's proposed solution has become the most popular cryptocurrency today, and in fact, most existing cryptocurrencies are evolutions of the architecture and algorithms presented by Nakamoto in that article. Since its publication and exponential growth, the scientific and technological community has dubbed the set of technologies and algorithms devised by Nakamoto as "blockchain" or "chain of blocks." Furthermore, it has begun to study the foundations of this technology because, for the first time, it enables reliable and secure transactions between two or more entities (people/machines) without the need for trusted intermediaries. This achievement is made possible through the aforementioned proof of work as a trust mechanism. Additionally, it provides users with protection for their identities and transaction data. Figure 2-5 shows how blocks, grouping transactions, are linked using hash technology in Bitcoin. Because each block contains the hash of the previous block, any attempt to alter the contents of a block would change its hash. This would break the chain, as the subsequent block would no longer have the correct previous hash. The transaction root, also known as the Merkle root, is a single hash value that represents all the transactions included in a block. Transactions are hashed and then paired together. Each pair of hashes is then hashed again,

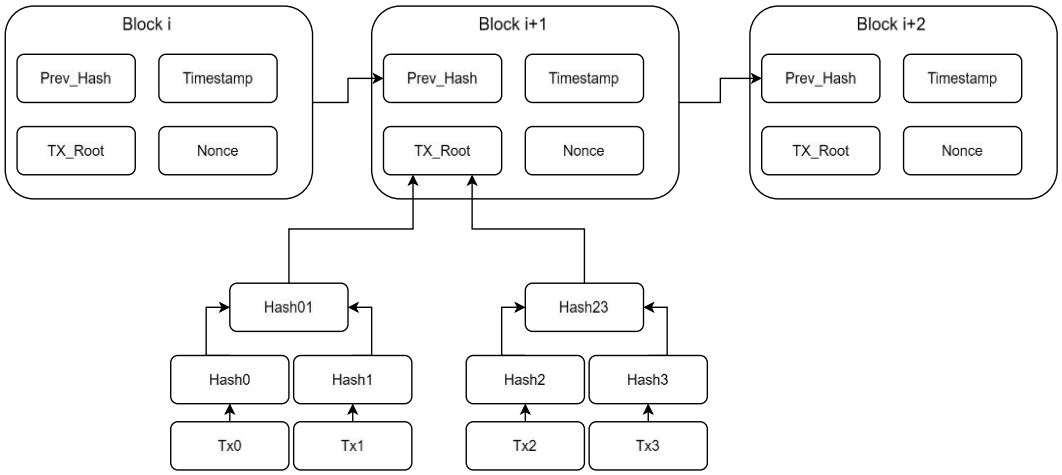

Figure 2-5: *Blocks linked by using hash technology in Bitcoin.*

and this process continues until a single hash remains, which is the Merkle root. The nonce is used in the process of mining, where miners compete to find a nonce that, when combined with the other contents of the block and hashed, produces a hash that meets certain criteria. Finally, the timestamp is a value included in each block that records the exact time when the block was created.

The cryptographic and consensus algorithms of blockchain ensure that transaction records are valid and immutable. This immutability is a crucial part of the trust mechanism. Often, an analogy is drawn to an accounting ledger where once an entry is recorded, it cannot be erased or duplicated (immutability). The technological pillars supporting a blockchain include:

- The blockchain and its distributed architecture/registry, that allows for decentralization of services through peer-to-peer networks, eliminating trusted intermediaries that could be single points of failure in critical systems.
- Cryptography, that is incorporated into the blocks of the chain, ensuring integrity and identification/authentication of participants.
- Consensus mechanisms, being verification and confirmation algorithms for transactions that guarantee system immutability.
- Smart Contracts, being machine code registered on the blockchain that enables automation of mechanisms and agreements among different nodes, thus providing a trust mechanism for relationships.

Different blockchain architectures can be used depending on the requirements of the use case. Although all current blockchains guarantee the integrity of the information, there are different blockchain architectures depending on level of permisioning:

- Private: not all data recorded in the blockchain is publicly disseminated. Only participants or users can access and query all or some of the transactions.
- Public: data recorded in the blockchain is available for anyone who desires to access it.
- Permissionless: data can be recorded in the blockchain by anyone.
- Permissioned: not everyone has permission to write in the blockchain. Only participants or uses can write and therefore generate transactions.

2.2 Related work

2.2.1 The “IoT-as-a-Service” business model

The concept of “IoT-as-a-service” has been introduced by some authors [17][18] as a feasible business model in which IoT devices are offered to clients for their usage in a “as-a-service” model. Yet, despite the fact that these authors have already identified and

defined the concept, they do not put the focus in analysing the technological challenges that it implies. Others [32] go a little bit further and describe four essential components in the IoTaaS: sensors and gateways, middleware, backend servers and output interface, but still lack in properly proposing a comprehensive model and analyze its implications. A search in Google Scholar using the keywords "IoT as a Service" or "IoT-as-a-Service" shows different works including these terms. However, none of them refers to the concept of IoT as-a-Service as considered in the present thesis, which presents a business model based on offering IoT devices on demand. Previous works refer to the intersection between the IoT and the cloud. For example, some authors [33] presented an IoT solution for public safety and disaster response (PSDR), while others [34] presented a formal model integrating IoT and eHealth, and others [35][36] referred to IoT-as-a-Service as a combination between IoT and cloud. Finally, other authors [37] propose an infrastructure for hosting IoT workloads in the cloud. The referenced works primarily focus on specific applications, but these contributions, while valuable, do not tackle the broader scope of IoTaaS as a suitable business model. The gap remains in providing a holistic approach to IoTaaS that addresses both technical and business aspects comprehensively.

2.2.2 Self-Sovereign Identity and IoT

From a research point of view, some studies have proposed the application of SSI to the IoT. First, some authors [38] thoroughly explained the SSI paradigm and presented decentralized identifiers (DIDs) and VCs for IoT. In addition, they analyzed the suitability of SSI for IoT, proving that it is a better method than the use of pretty good privacy (PGP) and X.509 certificates. Other studies have studied the applicability of the SSI paradigm to the IoT world. The Sovrin Foundation has conducted research on SSI and the IoT [39] and provides some useful insights for its application For example, some authors [40] introduced some SSI concepts and showed use cases for the industrial IoT, while others [41] briefly mentioned the intersection between SSI and IoT. These works show the "big picture", but they did not delve into any credential exchange protocols or technical details. Other authors [42] proposed the use of DIDs as identifiers for IoT devices and accurately studied the requirements for IoT devices to run an SSI-based identity management system, even proposing a proxy-based solution for extremely constrained IoT devices. Other proxy-based solutions were also proposed by other research works [43], in this case the IoT Exchange, to connect IoT devices with users. However, they do not suggest a specific VCs schema for the IoT and restrict themselves to just analyzing DIDs as suitable identifiers for the IoT. Other authors [44] presented an SSI protocol to trace the origin of IoT devices.

As explained before (see Section 2.1.1), SSI uses a decentralized registry that stores information such as DIDs and DID documents. This registry can be implemented using DLT, thus improving the overall security of the system by preserving the integrity and availability of the stored information. For example, some authors [45] proposed an SSI scheme based on IOTA [46] for its DLT backbone and applied it to a rental car use case.

The main problem of this proposal is that IOTA is not fully decentralized because it still depends on a centralized element that centralizes the consensus process. Similarly, others [47] also proposed IOTA as a DLT for implementing the SSI scheme. They accurately defend that IOTA is permissionless, has no transaction costs, and scales well; however, the main drawback is that IOTA is not fully decentralized. Furthermore, some mixed designs have been proposed in which SSI is used in combination with other elements in the same infrastructure [39][48]. Finally, there are studies that focus on specific applications. For example, some authors [49] proposed the use of a smart band for biometric identification following an SSI scheme, while others [50] particularized the SSI paradigm for Internet of Vehicles (IoV).

However, none of these works consider the particularities of the IoT-as-a-Service business model, such as the different actors who participate in it and the differences between usage and ownership of the devices. In addition, there is a lack of studies on the performance evaluation of SSI-based Identity Management Systems for IoT.

2.2.3 SSI technologies and frameworks

From an industrial point of view, there are several technologies and frameworks that enable the usage of SSI, including Hyperledger Indy (Sovrin) [51], uPort [52] (currently split in two projects: Serto, at this moment unreachable on the Internet, and Veramo [53]), Blockstack [54], Veres One [55], Jolocom [56], or Hyperledger Aries [57], among others.

Hyperledger Indy (Sovrin): Hyperledger Indy is a permissioned blockchain specifically designed for SSI. In this blockchain, anyone can read the content, but only authorized parties can write to the ledger. Indy supports VCs and is one of the most advanced frameworks in this field. It enables the creation of ZKPs, where a prover can generate a proof element for multiple attributes using different credentials from various issuers. From a credential mentioning multiple attributes, a user can selectively reveal only the requested attributes without providing the entire credential. Indy also employs derived predicates, allowing users to prove something via a boolean variable without revealing the exact attribute value. For example, a user can demonstrate they are over 18 years old without disclosing their exact birthdate. Both ZKPs and derived predicates enhance user privacy and prevent unnecessary data disclosure. The adoption of Hyperledger Indy is primarily driven by the Sovrin Foundation, an organization working to establish a global trust network that provides the legal and trustworthy foundation for SSI. Currently, the project has separated the blockchain network (Hyperledger Indy) from the agents communicating with it (Hyperledger Aries).

Hyperledger Aries: provides a shared, reusable, interoperable toolset designed for creating, transmitting, and storing VCs. On the other hand, Hyperledger Indy remains as a set of libraries and tools that allow to deploy a purpose-built distributed ledger for

decentralized identity. The key difference between them lies in their functionalities. Aries primarily covers the agent (client) part of a decentralized identity application that is intended to be agnostic to the underlying ledger/DIDs/VCs layer. Indy, however, is a decentralized identity implementation including support for a ledger (VDR), DIDs, and VCs. Initially, the idea was to move the agents code implemented in Indy to Aries, and so the first working versions of Aries use Indy underneath for the decentralized identity components. Over time, those components will become pluggable, and additional decentralized identity components will be supported. Thus, major parts of the indy-sdk will be deprecated, as they are now being migrated to Aries.

The Hyperledger Aries Cloud Agent [58] (ACA-Py) is a component within the Hyperledger Aries project. The ACA-Py specifically focuses on providing a framework for building and deploying cloud-based agents. ACA-Py operates within the second and third layers of the Trust Over IP framework [59], utilizing DIDComm messaging [60] and Hyperledger Aries protocols. The term cloud means that ACA-Py is designed to run on servers, thus not being intended for mobile device deployment. To interact with ACA-Py, developers need to implement a controller that communicates with it by sending HTTP requests and receiving webhook notifications. Due to this modular design, it is adequate for its usage in the IoT. By minimizing the piece of code that runs on the IoT (controller) it is compatible with constrained devices, and the SSI logic (agent) runs on a separate constraint-free server. It also allows developers to use any language of convenience for implementing the HTTP requests. Deeper information about the ACA-Py can be read in the official Github repository [58].

uPort: uPort provides a set of libraries and protocols for developing decentralized applications (dApps) centered around identity on the Ethereum platform [61]. Recently, uPort split into two separate projects: Serto and Veramo. Serto has become a proprietary solution, while Veramo remains open source. Veramo utilizes an interoperable identity layer composed of identity and messaging protocols. It also leverages smart contracts and VCs both on and off the blockchain.

Blockstack: Blockstack is a platform for building decentralized applications. It offers a decentralized naming and discovery service, allowing users to register human-readable and easily discoverable resources on the network. Blockstack also aims to provide a decentralized storage system where users can store their data without revealing it to any remote party. Blockstack uses a blockchain as the storage medium for operations and to achieve consensus on the order in which operations are written. Additionally, it operates a virtual chain. In the virtual chain, new operations are defined, and their metadata is stored on the blockchain itself. The blockchain is unaware of the virtual chain, storing only the metadata. Blockstack is not tied to a specific blockchain; currently, it uses Bitcoin [29]. Furthermore, there is a storage layer where actual data is stored. The data itself is not stored

on the blockchain. Regarding DIDs, Blockstack implements DID as a mapping between a user's registered name and their current public key.

Veres One: Veres One is a public blockchain specifically designed for decentralized identity and VCs. In Veres One, there are two types of DID-based identifiers. The first type is a cryptonym-based identifier, which is a SHA256 hash of a public key. It does not need to be stored on the blockchain and can be used as a pairwise identifier. The second type of DID is a UUID-based identifier, which can be used to store metadata on the blockchain. Veres One also supports off-chain VCs. The ecosystem utilizes accelerator nodes, which have elevated privileges in the network, allowing them to perform blockchain operations without proof of work. These nodes collect fees on behalf of the Veres One project, serving as the sole source of project funding.

Jolocom: Similar to uPort, Jolocom operates on the Ethereum platform. It provides a way to generate multiple identities from a master seed identity. Jolocom uses hierarchical deterministic keys, allowing the creation of secondary keys based on a known seed. From the seed, a pair of master keys is derived, acting as the primary key pair that can be used to generate various secondary key pairs. Secondary key pairs cannot be traced back to the master key pair. VCs can be private or public. Private credentials are stored on a device or in the credential owner's data storage option and can be shared with a verifier. Public credentials are stored on IPFS [62] and can be retrieved using the link information in the DID. Table 2-1 summarizes the main SSI implementations and their main features.

Indy/Aries present better combination between TPS, delay and cost against its alternatives. In addition, regarding the maturity level, both Veramo and Hyperledger Indy/Aries are the most advanced technologies for implementing SSI solutions. Therefore, Indy/Aries has been chosen for implementing the identity model presented in Chapter 4.

Table 2-1: *SSI implementations and features*

Platform	**Verifiable Credentials**	**Blockchain Type**	**TPS**	**Delay**	**Cost**
Indy	Yes (off-chain)	Own (permissioned)	10	Few secs	Free
Serto	Yes (on and off-chain)	Ethereum (permissionless)	15	15 secs	Ethereum Fee
BlockStack	No	Bitcoin (permissionless)	7	10 mins	Bitcoin Fee
Veres One	Yes (off-chain)	Own (permissionless)	200	Few mins	Depends on operation
Jolocom	Yes (on and off-chain)	Ethereum (permissionless)	15	15 secs	Ethereum Fee

2.2.4 Collaborative credentials

As far as we are concerned, no authors have still defined the concept of Collaborative Credentials (CC in what follows), being therefore a concept totally new at the literature. However, some of them have presented the concept of Group Membership Credentials [63], applied to the IoT. Here, the authors propose a system for proving the permanence to a group by different devices using VCs and CoAP as a communication protocol. Groups are identified by DIDs that allows to issue Group Membership Credentials to different owners of devices, who will "represent" the devices. The final goal is to make the verifier trust in a set of devices belonging to an owner. It is important to remark that a CC is not the same as Group Membership Credentials because a CC is created dynamically based on the collaboration of a group of participants and that is why they are called collaborative. Group Membership Credentials are created once and do not include a collaboration, nor any coordinator selection protocols. Similarly, although it has not been defined by the academia, we could define a Group Membership Credential as a credential identifying a group, but without the dynamism and collaboration factor of CCs.

While CCs are entirely a new concept, they make use of delegated credentials, which have already been studied by the research community. Regarding delegation, the first implementation was proposed by the authors in [64], and uses Growth-Sahai proofs, but these were very complex to implement. Other authors have proposed other delegation schemes [65], being also complex to implement. More recent delegation schemes [66] propose that a VC is directly passed to a new user that receives delegated permissions to issue new credentials, and therefore can now send copied VC signed by himself. This approach is also suggested by the W3C standard for VCs [24] in section C.5. Also, other strategies can be followed. Hyperledger Aries proposes creating a chain of trust where the VPs of the delegated credential are attached into the VC [67]. Chapter 5 compares W3C and Aries approaches for working with delegated credentials and presents some discovered issues when trying to implement a delegation with current technologies.

2.3 Summary of the chapter

Table 2-2 sums up the different references about SSI and CCs, grouped by topic categories, and presents the main conclusions extracted from them. These conclusions influence the decisions taken in Chapter 4 and Chapter 5.

Table 2-2: *Main articles grouped by topic categories and conclusions extracted from them.*

Type of reference	References	Conclusions
SSI technologies	[51][52] [53][54] [55][56] [57]	[51] and [57] technologies present a better maturity. This will be considered for the IoTaaS identity model and the CCs implementation (Chapter 4 and Chapter 5)
SSI standards	[23][24] [59][60]	Current standards do not consider CCs, which will be tackled in Chapter 5.
SSI use cases for IoT and applications	[38][39] [40][44] [45][47] [49][50] [63]	There are many in the literature, but we have cited some articles as an example for simplicity [38][39][40][44][45][47][49][50]. CCs are not covered in current literature. Therefore, Chapter 5 will present this use case. Group Membership Credentials [63] are similar to the concept of CCs, but they do not include the collaboration factor.
Delegation	[64][65] [66]	Although delegation has been studied in the literature [64][65][66], the present thesis will present some implementation issues with current technologies in Section 5.2.

The conclusions are as follows. The state of the art has provided a comprehensive overview of SSI and its main standards, especially in the context of IoT and discussed various SSI technologies and frameworks. Several key conclusions can be drawn from this research. SSI has been proposed as a suitable paradigm for IoT, offering advantages over traditional methods like PGP and X.509 certificates. Various studies have explored the application of SSI to IoT, including the use of DIDs and VCs. Some researchers have proposed proxy-based solutions for constrained IoT devices and the use of distributed ledger technology (DLT) to improve security. However, there is a notable lack of research addressing the specific requirements of the IoTaaS business model and performance evaluations of SSI-based Identity Management Systems for IoT. The state of the art also studies several SSI technologies and frameworks, with Hyperledger Indy/Aries [51][57] emerging as a preferred choice due to its combination of high transactions per second, low delay, and cost-effectiveness. Finally, the concept of CC is introduced as a novel idea in the field, distinct from Group Membership Credentials [63]. While CCs are a new concept, they build upon existing research on delegated credentials, whose current implementations face some challenges, which are explored further in Section 5.2.

This chapter has provided a comprehensive overview of the basic concepts and state of the art relevant to the IoTaaS business model and identity management in IoT environments. Several key conclusions can be drawn. First, the SSI emerges as a promising paradigm for addressing identity management challenges in IoT, offering advantages over

traditional methods like PGP and X.509 certificates. Second, the integration of SSI with IoT presents unique opportunities, particularly in enhancing security, privacy, and data integrity within IoT ecosystems. Third, while various studies have explored the application of SSI to IoT, there is a notable lack of research addressing the specific requirements of the IoTaaS business model. Forth, among the various SSI technologies and frameworks reviewed, Hyperledger Indy/Aries stands out as a preferred choice due to its combination of high transactions per second, low delay, and cost-effectiveness. Fifth, the concept of CCs is introduced as a novel idea in the field, distinct from existing concepts like Group Membership Credentials. CCs aim to address scenarios requiring collaboration among multiple entities, which cannot be fully addressed by standard SSI implementations. Sixth, current implementations of delegated credentials, which are related to CCs, face some challenges that need to be addressed for effective implementation in IoT scenarios. Finally, there is a need for further research and development in areas such as performance evaluation of SSI-based Identity Management Systems for IoT and in the definition and practical implementation of CCs.

Chapter 3.

The IoT-as-a-Service (IoTaaS) business model

In the IoTaaS business model, IoT devices are offered to clients for their usage in a "as-a-service" model. As discussed in Chapter 2, there is not much literature about the concept "IoT-as-a-Service". Although several authors have already identified and defined the concept, they fail in analyzing the technological challenges that it implies and there is a lack in properly proposing a formal model. Therefore, in this chapter, we analyze the IoTaaS business model to identify its main technological challenges and formalize its definition, which corresponds to the objectives A) and B) of the thesis (see Section 1.3). First of all, actors that participate of the IoTaaS are presented. Secondly, we identify the current technological challenges of the IoTaaS. Third, some security considerations are presented. Finally, we discuss the different applications for the IoTaaS.

3.1. Actors in the business model

Let us consider a person or entity denoted as *owner*, which possesses a set of IoT devices $\{D_n\ ;\ n\ \in \mathbb{N}\}$. Each *device* D_n can be "hired" by any interested consumer (person or entity) during a period of time. We will denote these consumers as $\{C_m\ ;\ m \in \mathbb{N}\}$. With the term "hired" we mean that the *device* is accessed for the data or services that it provides. For example, a *consumer* might be interested in the data provided by an IoT device, such as temperature or wind power readings, but also in using some services from this device, such as a connectivity test against other systems, or a notification in response to certain events. The amount of money each *consumer* will pay for the data or services will depend on the amount of time or resources that he will consume from these *devices*, as well as the type of the requested service, following a pay-per-use business model. It is also possible to

follow a subscription-based business model in which *consumers* pay a periodic amount of money (subscription) to access data or use services from *devices*.

This nomenclature can be extrapolated by considering several *owners*, $\{O_k\ ;\ k \in \mathbb{N}\}$, each one having its own set of *devices*. Each *consumer* C_m will be interested in obtaining services or data from *devices*, if and only if those devices meet some quality requirements. *Certification entities*, named as $\{CE_l\ ;\ l\ \in \mathbb{N}\}$, indicate that *devices* truly meet these quality requirements. Many actors, such as external auditors or manufacturers, can be *certification entities* because they can certify some attributes of the *devices. Certification entities* differ from Certificate Authorities (CAs). Certificate Authorities are companies or organizations that issue digital certificates (normally X.509) to different entities on the Internet. A *certification entity*, as described in the IoTaaS, is the actor who certificates some attributes of a *device*. This can be done by using digital certificates or by any other means. In the proposed identity model, this is done by using *VCs*, so we have decided to name it "certification entity", as this is a broader term than Certificate Authorities. The different roles of the business model are presented in Figure 3-1.

3.2. Technological challenges in the IoTaaS business model

Now, we briefly describe some of the main technological challenges that appear when facing a real implementation of the IoTaaS business model.

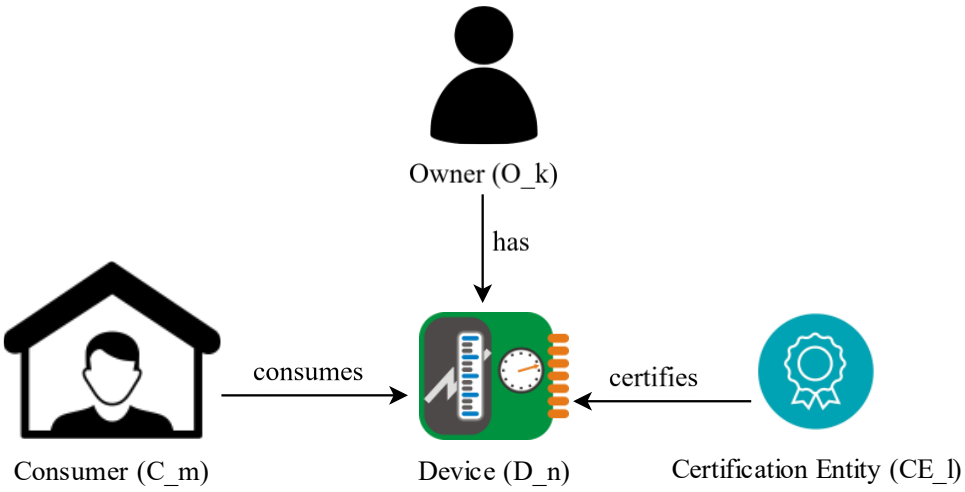

Figure 3-1: *Relationships among actors in the IoTaaS business model*

3.2.1. Service endpoint identification and availability

To begin with, one of the relevant aspects to tackle is defining how the services provided by *devices* will be offered to and accessed by different *consumers*. Regarding how to reach the services offered to *consumers,* it is quite straightforward that we need mechanisms to connect *consumers* with *devices*. One alternative taken from other as-a-service solutions is the use of a marketplace, that is, a mechanism to connect IoT devices with potential *consumers*. For our purpose, this marketplace is an interface where IoT *devices* are listed to be chosen by *consumers.* The marketplace also has the goal of preventing IoT devices from saturating consumers' requests, thus alleviating possible denial of service (DoS) attacks, which can be especially severe if these devices are resource-constrained. However, the final design of this marketplace could vary and become more complex depending on the requirements of the final implementation, as stated by some authors [68]. Thus, the marketplace appears as an optional but highly desirable element, because it enables more-constrained IoT devices to participate in the proposed IoTaaS.

3.2.2. Communication model and protocols

The communication model between devices, as well as between devices and consumers, and its subsequent protocols has been identified as an open research topic by several researchers. For example, some authors [69] proposed a privacy-preserving query scheme for IoT using homomorphic encryption, which can add an extra layer of privacy to the communications. Transferring data among IoT devices themselves or to the marketplace can be made by using IoT-specific protocols, such as MQTT-S, as usual in wireless sensor networks [70], or any other IoT-related protocol. In addition, secure versions of the MQTT and MQTT-S protocols might be considered, such as SMQTT and SMQTT-S [71], if the security levels require them. In addition, the chosen mechanism must be standardized, and owners will need to be aware of this standard before publishing their devices in the marketplace. As it will be shown in Chapter 4, the present research proposes the use of queries over SSI VPs to achieve this goal. Communication between consumers and the marketplace can be achieved using HTTPS or any other secure protocol.

3.2.3. Service semantic definition

There are some attributes of IoT devices that are important to be shown and queried in IoTaaS, namely:

- Identifier
- Data/service type
- Quality Certificates
- Cost

- Location and geographic scope
- Owner

First, it is crucial for the end user to know the data/service type provided by the IoT device, as the final goal of the IoTaaS business model is to provide data and services (e.g. actuators) from IoT devices. Following, quality certifications are also an important aspect because no user will pay for devices which are not able to prove their quality. As it happens in other business model, clients are becoming more exigent when it comes to the quality of their products and services, so this aspect is fundamental. Devices should be able to prove their quality, being mandatory to have a mechanism to ensure that the information provided by devices is legitimate. This means that there are quality certifications that have been issued by a trusted authority and they have not been revoked or tampered. Cost will possibly be the determinant factor for a user when choosing an IoT device and must be ideally calculated as a function of the previous factors and the demand and offer of the data or services offered by the devices. Other features that must be considered are the location and geographic scope. Some clients could have restrictions for accessing information generated in certain countries due for example to political conflicts. In this sense, it is important that each IoT devices provides information about its location and the geographic scope of their measures. The owner attribute defines the owner of the device.

3.2.4. Optimization

Another challenge to be analyzed is optimization. The range of possible scenarios to be analyzed in IoTaaS is large. Further studies on IoTaaS may study different optimization problems associated with the business model. For example, suppose that the expense of an IoT *device* D_n is E_n and its expected benefit is B_n. Both E_n and B_n can be functions of other variables, such as the type of data provided by the *device*, its location, or its quality. Then, the optimization problems can be formulated with different objective functions that allow us to know, for example, how to distribute the devices among different locations, how many quality certificates must the *devices* have, or the number and type of *devices* to be acquired to increase the profit.

Complex economic models can be built from this starting point, *for example*, models that consider the implicit cost of depreciation, which would impact E_n, or variable benefits $f(B_n)$ depending on non-stationary variables, *such as* season or demand.

3.2.5. Payment

A crucial enabling technology for the IoTaaS involves empowering 'things' to conduct transactions autonomously and universally with other 'things' without human involvement. This capability opens various possibilities but also raises some challenges. The means to facilitate these thing-to-thing payments remains uncertain at present. Current popular

payment solutions are not suited for handling the high volume of microtransactions due to limited capacity and high transaction costs. Additionally, credit card-based solutions pose security concerns, as sharing credit card information with devices can lead to further sharing with other devices or cybercriminals during transactions.

Micropayments have emerged as a promising solution to address these challenges in the context of the IoT. They involve the exchange of small amounts of currency in return for access to services or data, making them an ideal mechanism for facilitating transactions among IoT devices, as well as between IoT devices and users. By enabling devices to autonomously transact on a micro-scale, micropayments hold the potential to unlock new revenue streams and business models, as proposed with the IoTaaS business model, enhance data security, and empower individuals to have greater control over their IoT interactions. One of these revenue streams is the data monetization. IoT devices generate vast amounts of data and micropayments allow data providers to monetize their information on a per-use basis, creating a marketplace where data becomes a tradable asset. Another is resource sharing. Devices can share resources such as computing power, storage, or bandwidth with micropayments serving as compensation. Another business model is automated maintenance. IoT devices can pay for software updates, security patches, or maintenance services independently when needed, ensuring their continuous operation and security. Finally, micropayments enable energy-efficient operations. IoT devices can optimize their energy consumption by trading power with other devices, ensuring efficient energy utilization across the network.

While micropayments for IoT offer numerous advantages, they also come with their own set of challenges, which mainly are:

- Scalability: managing microtransactions across the IoT landscape can be complicated, so scalable solutions are essential to prevent congestion and delays.
- Security: the need for secure and tamper-resistant payment mechanisms is important to prevent fraud and ensure the integrity of transactions.
- Interoperability: micropayments must be interoperable across different devices, platforms, and ecosystems.
- Privacy: ensuring that micropayments do not compromise user or device privacy is a critical concern. Solutions that protect user data and preferences are crucial.

Previous research ([72][73][74][75]) suggests that blockchain technology can be employed for thing-to-thing payments. Blockchain-based micropayment systems solve three of four challenges: security, interoperability and privacy. Bitcoin [29] and blockchains exhibit several characteristics that make them suitable candidates for handling thing-to-thing payments. Firstly, they rely on decentralized peer-to-peer networks for

transaction recording and relay, allowing support for autonomous and numerous transactions. Another advantage lies in the ease and cost-effectiveness of creating new accounts, enabling each device to possess its own account, which can be generated in seconds without the need for internet access. Consequently, there is no central authority controlling these accounts, and they are not directly linked to individuals, but rather to the IoT devices themselves. Addresses, in technologies such as Bitcoin [29] or Ethereum [61], help to achieve privacy. Blockchain's distributed ledger ensures transparency and immutability of transactions. This is vital in an IoT environment where numerous devices are involved and trust between parties is essential. Being decentralized, blockchain-based systems are less susceptible to single points of failure, enhancing system reliability for continuous service provision. Finally, blockchain's borderless nature allows for transactions across geographical boundaries. This is advantageous for IoT services that operate on a global scale.

The IoTaaS business model requires an effective micropayments system for allowing the users to pay-per-use small amounts of money to the owners or even directly to the IoT devices; this is a huge consideration to be made, if the payments should go directly to the owner or through the IoT devices. These mechanisms should be carefully designed and suited for different scenarios. For example, payments could be made from consumers to owners (traditional scheme), or directly carried out to IoT devices, leveraging infrastructures such as DLT (e.g., blockchain) to manage and track transactions. With the latter solution, the process could be made without owners' intervention, using smart contracts as a method to automatically send the money when a certain balance is reached. In such an approach, it would be possible to decentralize the payment solution, thus improving the privacy of involved parties by using any type of blockchain address, such as the ones implemented in Bitcoin or Ethereum.

Each actor may be identified by a unique address, that allows him to receive and send payments. These addresses must be known in advance. The best option is to index this information in the marketplace, so *consumers* can obtain it when they query the marketplace searching for *devices* matching their needs. Each payment generates a blockchain transaction that is validated by the network nodes and registered in the immutable ledger.

3.2.6. Identity management

The identity problem must be determined by assigning identities to the different actors of the system. Some authors [76] provided a broad overview of the concept of identity and how it evolved over time, as will be explained in Chapter 4. To sum up, identity has suffered several transformations during the last few years, starting from an isolated and centralized model and later moving to a federated model. However, these models suffer from serious scalability and security issues. The last suggested model is the SSI [19]. As it will be

explained in Chapter 4 this model fits well for the IoTaaS and alleviates some of the main problems in previous schemes.

3.2.7. Collaboration

Another challenge that can be explored is the collaboration between a group of devices to meet a set of consumer´s requirements. Instead of requiring data from a single IoT device, there may be scenarios where a consumer requires data from a group of devices. This implies the necessity of creating groups dynamically and this topic will be addressed in Chapter 5.

3.3. Security considerations for the business model

The IoTaaS requires several security measures to ensure the privacy of the data being handled, the integrity of that information, the availability of data and services, and the authentication of devices. In summary, Table 3-1 incorporates some security measures with the expected positive impact for the IoT in terms of privacy preservation, integrity, availability and authentication.

Table 3-1: *Security requirements fulfilled by some proposed security measures*

	Privacy	**Integrity**	**Availability**	**Authentication**
Marketplace	No	No	Yes	No
SMQTT	Yes	Yes	No	Yes
DLTs	Partial	Yes	Yes	Only private
SSI	Yes	Yes	Yes	Yes
TEE	Yes	Yes	No	No
TPM	Yes	Yes	No	No

Regarding the proposed systems, the marketplace is not in itself a security measure, but it has been included because it can prevent DoS situations, and therefore provide an improvement in availability. SMQTT provides privacy, integrity and authentication through the application of Key/Cipher text Policy-Attribute Based Encryption (KP/CP-ABE) with Elliptic Curve Cryptography (ECC) to the MQTT protocol. DLT technologies provide integrity by design and higher availability as the number of nodes increases. Also, not all of them provide privacy, some of them being pseudo-anonymous, as mentioned above, and those that are private need an authentication process to interact with them. SSI improves the four stated security requirements [39][42] because it is an authentication tool that guarantees the integrity by design, improves the privacy through the use of ZKPs and selective disclosure and improves the availability by removing the identity silos. A TEE [77] is a secure area within a processor that ensures the confidentiality and integrity of the code and data loaded inside, because it guarantees that data cannot be replaced or

modified by unauthorized entities. Applications and their associated data are run separately from the rest of the system (privacy) with integrity preserving mechanisms; and likewise, TPMs [78] are generally employed to preserve the privacy and integrity of cryptographic keys.

Other issue that should be addressed is GDPR (General Data Protection Regulation). GDPR [79] requires protecting personal data from users to be managed without their permissions and includes some directives such as the right to be forgotten. In this case, for the IoTaaS business model, sensors should not have personal data from the owner or whoever, because if so, the user should be able to remove this data whenever required, and there are no formal guarantees for that to happen. Consequently, the easiest consideration is to avoid using personal data for the IoTaaS. There will be some use-cases where these data will be interesting for third parties, such as aggregated data from sensors to build statistics, which can contain personal data from users. In these cases, some considerations will need to be taken, such as using GDPR-compliant anonymisation tools for aggregated data or other means.

Another question that should be answered is about privacy and anonymity. Ideally, users paying for an IoT device should be anonymous to the rest of the people, so no-one should know who is paying for what device. The only important information is if one device is available or not to be used. In this sense, the system should provide a mechanism to preserve the privacy of the users but at the same time protect them from fraud. Furthermore, we can even imagine some scenarios where the owner of the device wants to keep his identity private, while in others the owner will want to be recognised as owner of his devices. For example, sometimes, some smaller providers or individuals will just want to earn an additional income for these devices, now become assets, without unveiling their identity. The chosen identity management mechanism should provide with means to both be able to protect the identity of the owner of the devices or be recognised.

3.4. Applications

Although this concept can be applied to a myriad of scenarios, we present here a concrete use case to make these concepts easier to be understood by the reader. In this case, a person is a researcher who wants to use the information provided by some sensors located in a coastline gathering information about tides. These sensors are property of a research company C, which has already paid for these devices and invested time, money and human resources in their setup (configuration, location, etc). This company C can earn an extra money by sharing these devices during a period with person P, who finds the information from these sensors very useful for his own research. This simple example arises some considerations that need to be taken. For example, this researcher needs to be sure that the IoT device has quality certifications certifying that the measures they offer are accurate

enough. The IoTaaS works here as a low-cost generator of datasets, where researchers can benefit from other´s devices to generate their own data for their own research.

As another example, let's consider an IP camera in a city recording images and videos of the traffic inside that city. Again, a group of researchers from the Government is interested in the aggregated traffic data to make predictions about traffic jams in rush hours. These researchers want to have access to a group of cameras during a period. This simple example arises some considerations that need to be taken. This example shows another necessity, which is that users need to be sure that the taken measures have not been tampered, and that data privacy is guaranteed during the whole process. Furthermore, it would be catastrophic if this camera would be substituted by another one controlled by an attacker, who might send fake information to the researchers, so a solid identity management is needed for the users (researchers in this case) to be sure that they are interacting with the legitimate camera, and that this camera is really operational. Note that, when talking about identity of an IoT device, the different attributes of the camera are part of its identity, together with its quality certifications. Only few people would pay for using an IoT device without checking that this device has quality labels from the manufacturer, and it is properly certified. As the IoT is an extensive world, with a myriad of devices out there, this research field is highly related to Big Data, where statistics and predictions are made over enormous datasets. Also, machine learning operations can be made on the gathered data to extract useful information. In the example, the information provided by the IP cameras can be used to improve the Traffic Monitoring and Management Systems with real time information.

As a last example, the information provided by the IoT devices can also be used to perform market research operations. For example, some agents could want to get information from different IoT devices located on the beach to know how many people is spending their holidays there, which can help these agents to detect potentially undervalued areas where they can invest in.

Managing the identity of groups, rather than just individuals, is crucial in scenarios where collective data and actions are involved. For instance, in the case of researchers accessing data or services from multiple IoT devices that need to cooperate each other. Managing group identities helps in scenarios where multiple devices work together to provide aggregated data, instead of individual outputs. Use cases covering the necessity and implementation of group identity management are presented further in Chapter 5.

3.5. Summary of the chapter

This chapter has presented a comprehensive analysis of the IoTaaS business model, formalizing its definition and identifying its main technological challenges. The IoTaaS model introduces several actors: owners of IoT devices, consumers who hire these devices,

and certification entities that validate device quality. Major technological challenges for implementing IoTaaS include: i) service endpoint identification and availability, ii) communication models and protocols, iii) service semantic definition, iv) optimization of device distribution and pricing, v) payment systems, particularly micropayments, vi) identity management. Security is paramount in IoTaaS, so further chapters will explore the identity management challenge, which is crucial to achieve security. Furthermore, the IoTaaS model has diverse potential applications, from research data collection to traffic monitoring and market research. The concept of CCs, as is explored in Chapter 5, emerges as an important aspect for scenarios requiring cooperation among multiple devices.

Chapter 4.

Identity in the IoTaaS

The present chapter intends to present an identity model based on SSI for the IoTaaS, which corresponds to the objective C) of the Thesis (see Section 1.3). As discussed in Chapter 2, after surveying the state of the art regarding SSI and its applications for the IoT, there are no studies discussing the particularities of the IoTaaS business model with respect to the different actors who participate in it and the differences between usage and ownership of the devices. Therefore, the present chapter deal with these aspects, having the following content. First, and, for introductory purposes, the evolution of the concept of identity is presented. Secondly, we explain the benefits of applying SSI to the IoT. Finally, we propose an identity management system for the IoTaaS based on SSI and include some security considerations. This identity management system will be further tested in Chapter 6.

4.1. The evolution of identity

A Service Provider (SP) is an entity that provides services to users. In the context of identity and access management, a SP is typically a website or application that relies on an external Identity Provider (IP) to authenticate users. Therefore, an IP is an entity that creates, maintains, and manages identity information for users and provides authentication services to other applications. For example, when you log into an online service using your Google or Facebook account, that online service is the SP, while Google and Facebook are acting as IPs.

Following the classification made by authors in [76], traditionally, the identity problem on the Internet has been dealt with using what is known as the *isolated model.* In the *isolated model,* the SP and the IP are the same entity, being considered a reliable party, offering access by a single security domain. Users need to trust these components to store

their identities, thus making them become honeypots for attackers. A risk in the isolated model is that IP might easily run away with all the identities, or prevent the users from using the service, leaving them completely unprotected. In addition, it acts as a single point of failure (SPoF) because the entire system relies on it.

With the emergence and popularity of services on the Internet, the isolated model became unmanageable because of the huge number of identities to be managed by a single SP, so a *centralized model* was developed to solve this issue. In this model, the SP and IP are not the same entities. Instead, different SPs share the same IP. However, it does not solve the previous issues, because the users still need to trust an external entity and the IP remains as an SPoF, thus becoming a honeypot for attackers.

Finally, the *federated model* differs from the centralized approach in the creation of trust relationships among different IPs. It allows the establishment of networks of IPs, improving the flexibility and overall trust of the system. The advantages of federated identification are diverse for users: 1) thanks to the Single Sign-On (SSO), users can use the same personal identification to access different services across various domains, 2) when logging in to a federated service, users can move to any other service within the federation without needing to reauthenticate, 3) logging out of one federated service automatically logs the user out of all federated services, 4) users can access a federated service without manually creating an account, because the necessary attributes are automatically generated based on the identity provider's release, 5) federated systems request user authorization to share necessary attributes with the target service.

However, the subjacent problem still remains, *that is*, users need to trust the IP, which can still invalidate the user´s identity at will. In addition, scalability issues are not solved; as the number of identities increases, the IP needs to properly manage them; at the same time, it also becomes a honeypot for attackers and an SPoF. In this context, and in the face of this issue, there arises a need for a decentralized model that restores control over personal information and identity to the user. Such a model would eliminate the information silos that are appealing to malicious actors. In this way, the user themselves would have custody and full control of their data, sharing only what they desire and when they choose to do so. The SSI paradigm emerges in this context, intending to solve the issues associated with the previous identity models. Its main feature is that the user is the owner, manager, and exclusive custodian of their identity and data. As a result, data owners are the only ones who can use the data to share it with third parties. The SSI concept has been deeply explained in Section 2.1.1. In light of all these features, it can be appreciated that decentralized identity offers numerous advantages over other digital identity alternative.

4.2. Justification of the use of SSI for the IoT

While the potential of SSI in various domains and sectors is undeniable, its integration with the IoT ecosystem presents a unique and compelling opportunity. The IoT, with its sprawling network of interconnected devices, generates an enormous volume of data that holds immense value for individuals, organizations, and society. Getting value of this data is precisely one of the objectives of the IoTaaS. However, as stated in Chapter 1, security is crucial for the IoT and there is a lack of robust identity management mechanisms within the IoT ecosystem, which introduces significant challenges related to security, privacy, and data integrity.

Privacy itself improves when using DIDs as identifiers for IoT devices [41] Furthermore, the traditional identity management schemes need what are known as "identity silos". These identity silos are a perfect honeypot for attackers because they usually store the identities of thousands of devices, which is a sweet information to be stolen by an attacker, so, an identity management system which can remove these identity silos should be considered. Also, it simplifies the work done by organisations because they will only need a single connection with an identity layer on the internet, being this advantage more appreciable when it comes to managing thousands of IoT devices. Finally, the cost of managing identities is also reduced with SSI because of the elimination of intermediaries. By eliminating identity silos and giving the user the power to manage his own identities, intermediaries are highly reduced, which implies an important saving in costs.

Another important advantage with SSI is interoperability because it enables an interoperable way to manage the identity of different devices. As it has been seen before, diversity is one of the features which characterise the IoT, being these devices made by different manufacturers. Consequently, by designing an interoperable identity management system we are providing a solution to these diversity issues (different manufacturers making different devices) which ultimately leads to the prevention of identity theft.

Unattendance feature can also be overcome with an SSI scheme. Following, there is an excellent example of an unattendance security risk applied to a connected car and how SSI can solve this issue. The different individual parts within a connected car can authenticate against the CAN bus using SSI credentials to prove that they have not been tampered. For example, the car owner, Bob, is notified that someone has tried to tamper his car to commit a further crime against him, all thanks to the use of SSI. At the same time, the same strategy can be applied to prevent an interdependence attack. In this case, the IoT device can broadcast an SSI prove of his current non-altered state to other IoT devices interdependent with him to prevent this kind of attack vector.

To sum up, based on the previous model of features [2] described in Section 1.1, we represent in Table 4-1 the features that are covered by the SSI:

Table 4-1: *Map between the features model and SSI*

Feature	**Covered by SSI**	**Not covered by SSI**
Ubiquity	X	
Mobility		X
Unattendance	X	
Constriction		X
Interdependence	X	
Myriad	X	
Diversity	X	
Intimacy	X	

First of all, SSI addresses the challenges of ubiquity and myriad devices by eliminating identity silos, which become increasingly risky as the number of connected devices grows exponentially. Second, SSI can be seamlessly integrated into unattended devices, enabling autonomous authentication without human intervention. Third, by allowing devices to authenticate each other before establishing communication, SSI significantly reduces the risk of interdependence attacks. Forth, SSI proposes a standardized authentication method, providing a unified solution for the diverse range of IoT devices in existence (diversity). Finally, SSI enhances privacy protection through the use of Zero-Knowledge Proofs (ZKPs) and selective disclosure, allowing devices to present only the necessary information for authentication. However, SSI does not have any impact in mitigating the mobility nor the constriction features.

As it can be deduced from the above, SSI also provides many benefits against the rest of the models for identity management: isolated, centralized, and federated models [12]. First, it removes the need for "identity hubs" by empowering the users to manage their own identities. Second, and more importantly, when it comes to the IoT, it scales better as a consequence of removing these identity hubs, which is especially true in IoT environments [76], where ubiquity or the existence of a myriad of devices are essential features to be considered. This fact clearly complicates the identity management process using traditional approaches. Finally, some authors [39] recognized some additional benefits when applying SSI to the IoT, which can be summarized as an increase in revenue, cost, and risk reductions. This study also analyzes which cybersecurity threats affecting the IoT can be mitigated using an SSI-based IdM. One question to be addressed is whether IoT devices have sufficient resources to run an SSI-based identity scheme. Some authors [42] precisely analyzed the minimum requirements that IoT devices must have in order to run an SSI solution and conclude that most devices are able to do it. In addition, if this is not fulfilled, they propose a proxy-based approach for extremely constrained devices.

4.3. A proposal of identity management system for the IoTaaS

After discussing why SSI is the best solution candidate for a proposal in the IoT identity management problem, we present a novel SSI system for the implementation of an identity solution for the IoTaaS. This system was designed following the guidelines of the W3C standard for DIDs [23] and VCs [24]. Hence, in the proposal, we assume that all the processes, for example, creation, storage, revocation, and presentation of credentials, follow these standards and recommendations.

We divide the proposal for the identity management process in IoTaaS in three subprocesses. The first one is called identification and initialization, and it is partially represented in Figure 4-1. The second subprocess, shown in Figure 4-1 partially and in Figure 4-2, is called the Credential management subprocess, and the third one, shown in Figure 4-3, is called the service consumption subprocess. In the following, we detail them.

Before describing the proposal for identity management in IoTaaS, it is relevant to indicate that all the aforementioned actors in IoTaaS, that is, *certification entity* (*owner*, *manufacturer*, or whoever), *device*, and *consumer*, can play different roles from the identity management perspective, that is, *issuer*, *holder*, or *verifier*, depending on the stage of the process in which they participate. In what follows, we clarify this.

4.3.1. Identification and initialization

The first relevant aspect of the architecture is the identification of participants. Here, each *device* D_n has its own DID (or a set of them) to be identified and its own private and public keys are stored in its wallet for signing VPs. This DID allows the identification of IoT *devices* while preserving their privacy. The DID and its corresponding DID document are created and, optionally, registered in the VDR, following the guidelines provided by some authors [41] and are represented in Figure 4-1. Some IoT scenarios will not make possible to have a VDR, so this step will not be done. This may occur in large scale deployments where the number of devices is huge, thus making unfeasible the registration of numerous DIDs and its associated DID Documents in a VDR. For example, a DID of the kind *did:key* can be used, thus eliminating the need of a DID Document and its registration in the VDR. However, it is recommended to use one, if possible, as discussed in Chapter 2. In addition, *VPs* are signed by each D_n, so no tampering can be made by a

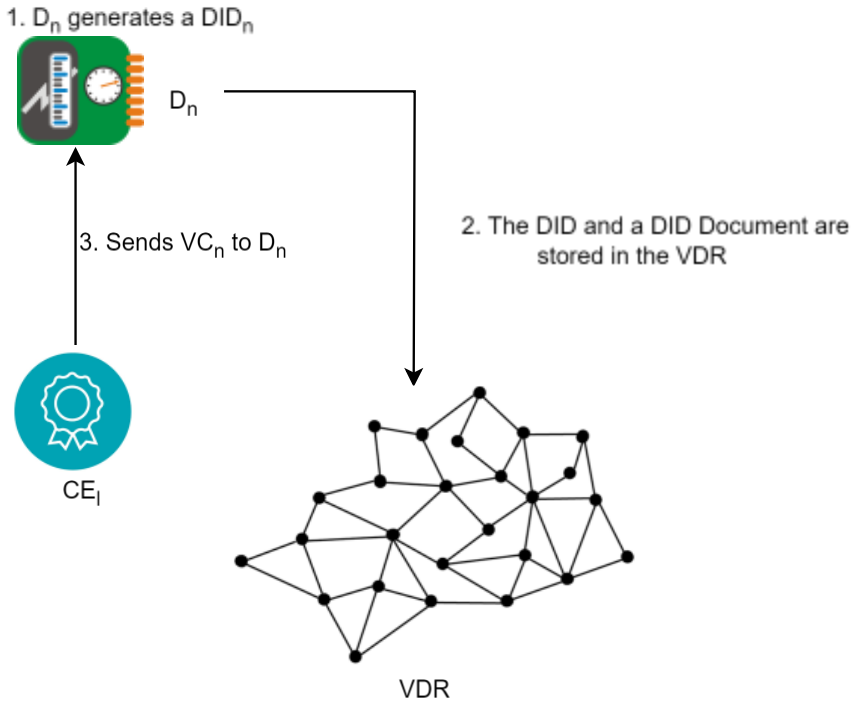

Figure 4-1: *Identification and verifiable credential management subprocess*

malicious marketplace because *consumers* can check the authenticity of the *VPs* by using a VDR

Furthermore, it is required for the *owner* to enable his *devices* to participate in IoTaaS. This can be done either manually (*the owner* creates the *devices´* DIDs and DID documents) or automatically (*the owner* sends a one-time token to his *devices* and they create their own DIDs and DID documents). Ultimately, these DID and DID documents will be optionally registered in the VDR if there is an existing one, and, consequently, these *devices* will be enabled in the IoTaaS business model.

Similarly, each *consumer* C_m and *certification entity* CE_l perform the same process of generating their own DIDs. These DIDs, together with their corresponding DID documents, are stored in the VDR after their creation. Differently from the IoT, the number of *consumers* is not expected to be big, so there will always be recommended to have a VDR.

Another important element will be part of this scheme as a solution for the *service endpoint identification and availability* challenge: the marketplace. It acts as a directory service for the location and search of available *devices* by *consumers*. In addition, it behaves as a storage for VPs, where *devices* store their proofs for their VCs. By doing so,

consumers can check these *VPs* directly from the marketplace, avoiding saturation of *devices*; in other words, it acts as a cache for *VPs*.

Differently from *VPs*, VCs are not stored in the marketplace, so *devices* can maintain control of the information sent to the marketplace via VPs, which can contain ZKPs, thus ensuring privacy as needed. Note that the marketplace cannot generate ZKPs because this element is not the original *holder* of these VCs, so only the *devices* can generate them. Note that there is a trade-off between security and performance because ZKPs could improve the privacy of *devices*, but their use implies that these *devices* would need to answer queries and verification requests from *consumers*, as well as generate ZKPs, which clearly impacts performance.

Regarding the centralization of this component, the marketplace is preferably implemented as a distributed system, thus avoiding becoming a single point of failure. Centralization can be avoided by using decentralized technologies to implement the marketplace or even using multiple marketplaces [80]. Using a blockchain ensures transparency through smart contracts and adds an extra layer of security to the marketplace. However, blockchain technologies may be troublesome to implement the storage of *VPs* in the marketplace because in this case *VPs* would not be able to be removed from it, which goes against the right to erasure defined in the GDPR [79]; but some alternatives can be followed, such as using off-chain storage for the *VPs* but keeping its hash on the blockchain. Other distributed alternatives, such as distributed versions of the LDAP [81] can be used to implement the marketplace.

4.3.2. Credential management subprocess

This subprocess deals with two different aspects: *i)* how VCs are issued and *ii)* how these credentials are verified. Regarding credential issuing, we assume that a set of *certification entities* issues certain VCs to a *device* D_n. Hence, from the point of view of identity management, the *certification entities* act as *issuers* and the *device* is a *holder*. Note that these VCs are part of the identity of *devices* and contain information about each D_n , such as, for example, what quality certificates it has. Hence, a device´s identity is composed of all its VCs. For example, a *manufacturer* (acting as a trusted *certification entity*) could certify the waterproof property for all his *devices* issuing them a VC for this property. In addition, these *devices* may have a different *owner*, O_k. The *owner* could also act as a trusted *certification entity* by sending VCs for specific features, such as ownership. Figure 4-1 represents this VC issuing process by a *certification entity* after DID creation and registration.

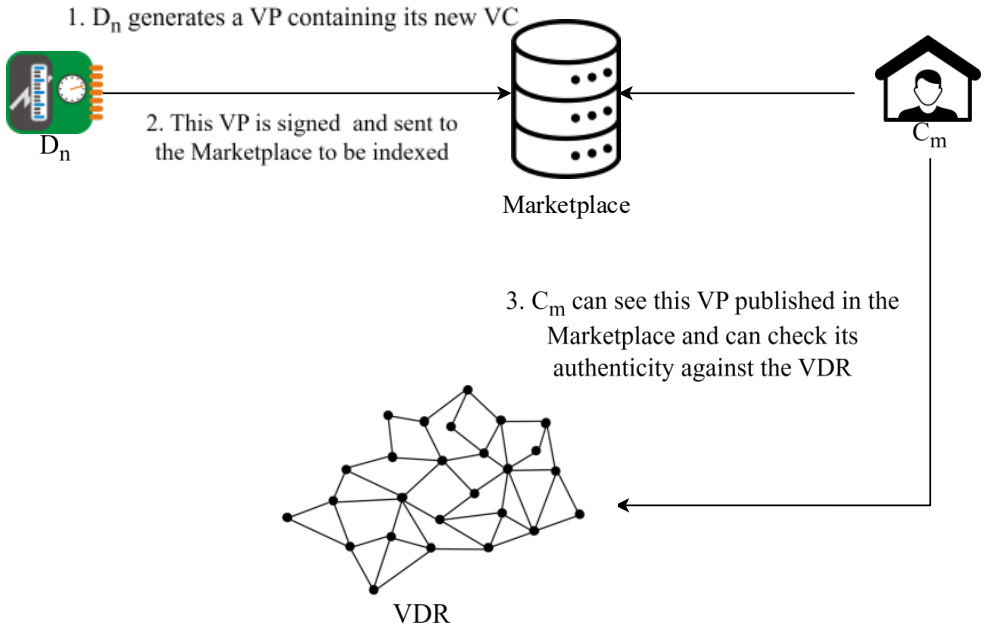

Figure 4-2: *Verification process*

With respect to the verification process, *consumers* $\{C_m\ ;\ m \in \mathbb{N}\}$ in the IoTaaS model act as *verifiers* from an identity management process, while *devices* $\{D_n\ ;\ n \in \mathbb{N}\}$ play the role of *holders*. Now, each *device* D_n generates a VP from its set of VCs, so anyone can check that the selected *claims* are valid. This VP is signed by the *device* and sent to the marketplace to be indexed. Consequently, *consumers* can query the marketplace to retrieve information about each *device* to determine whether it satisfies their needs by checking its authenticity against a trusted VDR. This process is illustrated in Figure 4-2.

4.3.3. Service consumption subprocess

The *service consumption subprocess* is illustrated in Figure 4-3. First, a *consumer* C_m queries the marketplace to select the most suitable *device* for his/her needs. Remember that in the *Credentials management process, devices* have sent a VP with all the required VCs to the marketplace containing their identity attributes, so the marketplace can perform queries over the attributes in the VCs and return a list with the *devices´* DIDs that match the *consumer* query. Once the *consumer* has selected an available *device* D_n, he requests access to this *device* and sends it a micropayment. As explained in Chapter 3, micropayments can be made either to the *owner* or to the *device*, and the latter is preferable in terms of *the owner´s* privacy. Then, if D_n agrees with C_m request, it sends a signed message to the marketplace. The marketplace then has information about whether D_n is available for use, so it can include the *device* in his list of unavailable devices. *Consumers*

only have to check this list provided by the marketplace without interrupting the normal operation of D_n. This prevents other consumers from selecting this device. Note that some devices may allow several concurrent consumers, while others may not. This entirely depends on the device configuration and capabilities and is beyond the scope of this study. At the same time, D_n sends a *consumer credential* to C_m so that he can connect to the IoT *device*, while this credential is valid. However, the use of *consumer credentials* as an authentication method adds security to the process while may penalize performance, because *devices* must verify this credential each time an authentication attempt from a *consumer* takes place. However, in Chapter 6, some performance measures are provided to show that the access mechanism based on *consumer credentials* does not actually imply performance degradation unless the *device* is extremely constrained. In this case, alternative mechanisms could allow controlled access to devices, such as the issuance of temporary access tokens. Further research can study the optimal combination between tokens and *consumer credentials* to achieve the best performance-security ratio. Another interesting research topic could be the generation of access tokens based on a given VC.

A *consumer credential*, which serves as proof of authorization, is a special kind of VC that allows a *consumer* to access data or services from an IoT *device*. This third subprocess (see Figure 4-3) involves the *consumer* getting a *consumer credential* for consuming data or services from an IoT *device*. The *consumer credential* has its own attributes, being relevant the *issuanceDate*, allowing anyone to know when it has been issued and the

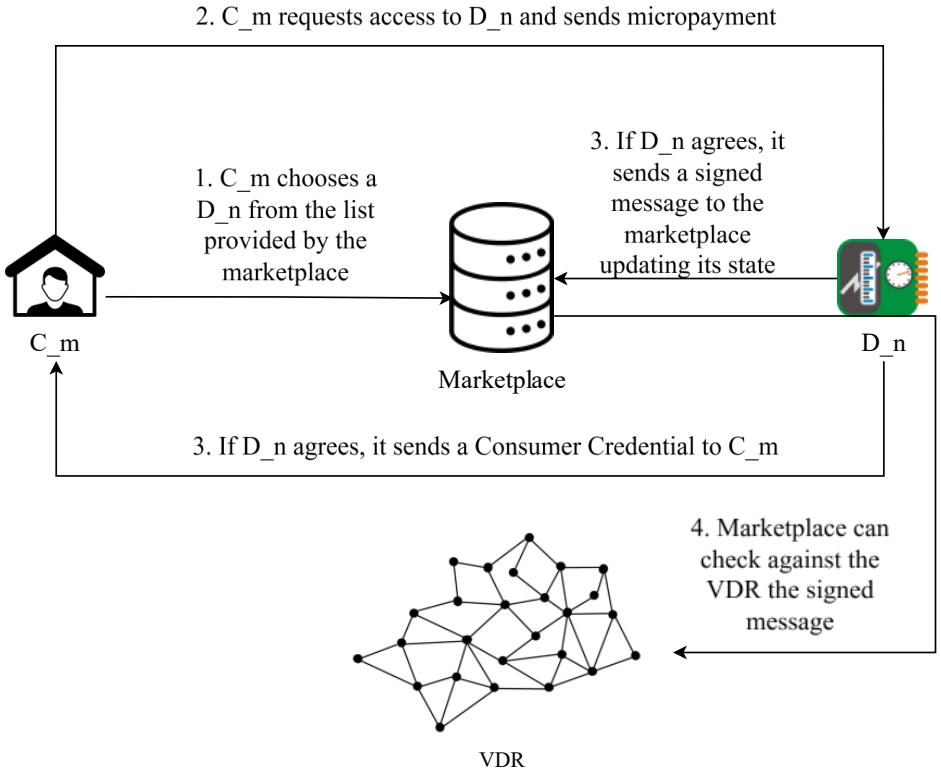

Figure 4-3: *Service consumption process workflow*

expirationDate, which contains the date when the credential is no longer valid, both specified in the W3C [24]. These two dates define the timeframe during which the *consumer* will be able to consume data from the IoT *device*. *Consumer credentials* cannot be modified, so any update in the credential, such as a new *expirationDate* implies a new issuing process. In this subprocess, the *device* plays the identity management role of an *issuer*, while the *consumer* acts as a *holder*. In addition, the *device* acts as a *verifier* for its *consumer credentials*, as it can verify this credential issued by itself to the corresponding *consumer* every time he wants to consume data from it.

A pay-per-use payment method would require assigning a timeframe long enough to contain the timeframe during which the *consumer* consumes data from the IoT *device*, and also a revocation method. In addition, it is important to have a mechanism for the IoT *device* to revoke the *consumer´*s access in case of payment failure or breach of contract. In addition, some payment methods, such as the pay-per-use, may require the revocation of *consumer credentials*, where the *consumer* pays for the number of resources consumed by the *device*, instead of the time. Focusing on the revocation, there are no differences in this procedure compared with the standard SSI revocation process described by authors in [24]. Revocation is made by using a *revocation list* recorded in the VDR, so the *holder* will no longer be able to prove that he has this valid credential anymore. This approach has immense value for the *issuer*, because the number of IoT *devices* is enormous and the *issuer* does not need to answer requests for every party desiring to check whether a VC issued to a *holder* is valid or not, he just needs to update this *revocation list*.

4.3.4. Security considerations for the identity model

Some security and privacy considerations about the proposed identity model are given. First, we have already discussed the privacy of the exchanged device attributes in Chapter 3. These can be protected by devices by generating zero-knowledge proofs when compiling VPs, before sending them to the marketplace.

Another aspect to be considered is related to data aggregation. Data aggregation occurs when *consumers* aggregate information from the same *device* when asking for different VPs to the same device. In other words, with each presentation request, a *consumer* will learn additional information from the same *device*. This is a difficult privacy problem to address, but ZKPs can solve this aggregation problem by hiding unnecessary information during subsequent presentation requests. Solutions to this problem are often policy driven, so if a device wants to avoid aggregation in his data, he could indicate it in the VP he generates, so the *consumer* knows that he or she cannot keep making presentation requests. *Devices* could also block requests from the same endpoint to prevent this scenario.

Furthermore, other security risks must be considered, even if ZKPs are used. Different side-channel attacks can be used to retrieve useful information about the subject of VCs.

For example, there are mechanisms on the Internet to track individuals, such as cookies, web browser fingerprinting or position information. The proposed scheme cannot prevent the use of these tracking technologies if they have been installed in the *consumers* or *devices*. Hence, their use must be avoided in critical scenarios.

Another risk could appear when *certification entities* include links in VCs. A VC is tampering-protected, but the content outside this credential is not, so links can cause harm as the linked data can be modified by an attacker. This can be avoided by using content-integrity protection for links to external data, such as storing the content in IPFS [62]. The W3C standard [24] describes about how to protect against these and other similar attacks in Sections 7 and 8.

4.4. Summary of the chapter

This chapter has presented an identity management system based on SSI for the IoTaaS business model, addressing the identity management challenge identified in Chapter 3. SSI offers significant advantages over traditional identity models for IoT, addressing challenges related to some features that characterize the IoT, which are: ubiquity, unattendance, interdependence, diversity, and privacy. The proposed identity management system makes use of *consumer credentials* to facilitate secure access to IoT devices and their services. An additional component, the marketplace, is introduced to act as a directory service and storage for VPs, improving scalability and preventing device saturation.

This page was intentionally left blank.

Chapter 5.

Collaborative Credentials

The present chapter intends to make a deep dive into the concept of Collaborative Credentials (CCs), which corresponds to the objective D) of the thesis (see Section 1.4). As discussed in Chapter 2, this is a novel contribution not found in the current state of the art. CCs are an evolution of the identity problem, presented in Chapter 4, in such a way that what has been analysed in Chapter 3 is extended here to support additional IoTaaS use cases that require collaboration between a group of participants.

This chapter is structured as follows. First, an introduction is written so the reader can understand what CCs are, where they come from and their implications. Secondly, the delegation problem is studied from an implementation perspective as a problem related to CCs. Third, the need for collaborative CCs is analysed, as well as some of their applications in different scenarios, including the IoTaaS. Fourth, a formal model for CCs is defined. Fifth, a lifecycle for CCs is described. Sixth, a security analysis of CCs is performed. Finally, the chapter proposes an implementation for CCs. This implementation will be further tested in Chapter 6.

5.1. Introduction

Normally, one VC contains claims that are related to a single entity. This implies that this entity can proof its ownership by issuing a VP to the verifier. However, there are scenarios where no single and isolated entities but a group of them might need to collaborate to achieve a common goal. In other words, each member of the group has his own capabilities (certified with traditional VCs) but all of them must collaborate to gather all the capabilities required by the *verifier*. That is, a set of capabilities owned by the whole group. This situation requires that members of a group stablish communications,

collaborate, and come to agreements about what capabilities will be shared. Regretfully, these mechanisms and procedures are not covered in current standard SSI flows, such as in the W3C [24] standard for VCs. For this reason, we propose in this Chapter the concept of Collaborative Credentials (CCs), to extrapolate the concept of single user VCs to credentials that represent the capabilities of a group of participants and are created upon collaboration of the members of the group.

CCs are a type of VC designed for scenarios requiring collaboration among multiple actors, where a verifier needs to periodically confirm the group's capabilities. Unlike standard VCs, which are static and unsuitable for dynamic group scenarios, CCs allow a designated *coordinator* to represent the group, acting as a single point of contact with the *verifier*. The *coordinator* would need a unique type of VC that accredits the group's collective capabilities, which does not currently exist, and this is the CC. This approach addresses performance, privacy, and scalability issues by avoiding individual polling of *group members*. Section 5.3 will shed more light to these aspects. Initially intended for IoT applications, CCs enable devices to collaborate in real-time to meet requirements that cannot be fulfilled individually, thus unlocking new applications beyond the capabilities of standard SSI.

The concept of CCs makes use of the concept of delegation. Delegated credentials refer to credentials which allow the user to delegate some of his attributes to another user, being a term recognized by the W3C standard for VCs [24]. With delegation, the original holder can designate and therefore authorise another person to hold his credential on his behalf. This can normally happen, for example, when a patient is too ill to take a prescription, and he designates a friend or family member to go to the pharmacy on his behalf. Delegated credentials are commonly formed by two parts: the delegated credential and the authorization, by which the original holder authorizes another holder to present his credential. How this authorization is provided depends on the technology that implements the delegation. For example, Hyperledger Aries proposes its own delegation scheme by storing all the information that the *verifier* may need to verify the delegated credential in a field named *provenance_proofs*, encoded in base 64. Following the guidelines of the Aries RFC [67], the *verifier* will ask for a field *provenance_proofs* where the VPs of delegated credentials are included, encoded in base64. This aspect will be discussed later in detail.

Collaborative Credentials enable collaboration within the IoTaaS business model, as identified in Chapter 3. Therefore, with CCs, it is possible to cover the collaboration between a group of devices to meet a set of consumer´s requirements. This will be tackled in the present Chapter.

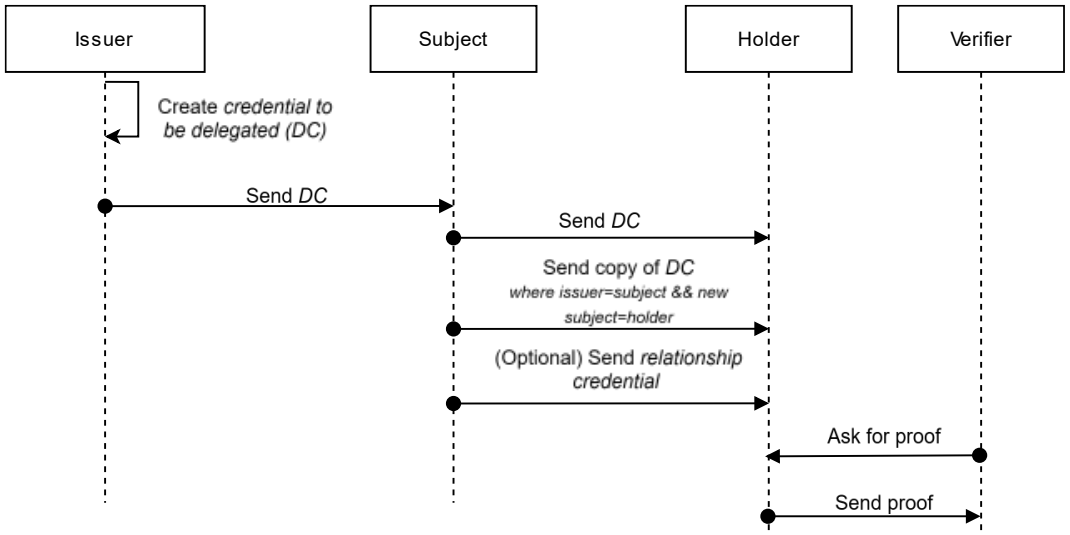

Figure 5-1: *W3C proposal for delegation*

5.2. Current implementation issues with delegation

It is important to consider for this thesis the existing current limitations when implementing a credential delegation with existing technologies, as CCs use delegation. Although the VCs model proposed by the W3C standard defines certain high-level guidelines that can be used for creating delegated credentials, it does not go deep into *i)* providing implementation guidelines for the issuance of delegated credentials (as it will be analysed, most of technologies do not currently support this approach) and *ii)* detailing how to deal with effective verification of delegated credentials.

In what follows we analyse the W3C standard [24] and the Aries [67] approach for building and working with delegated credentials. The W3C standard specifies how delegation must be implemented, while the Aries RFC provides a proposal for delegated credentials, compliant with the W3C.

W3C standard. The W3C standard proposes expressing the relationship between the delegator and the delegate either using a *relationship credential* (RC), issued by the delegator to the delegate, or in a separated attribute within the delegated credential (DC). This RC expresses a relationship between the delegator and the delegate, this authorizing the delegate. Hence, the *verifier* should accept a VP if it includes a DC and the corresponding relationship between the delegate and the delegator, which can be represented through a RC or within a copy of the DC in a separated field. The DC is transferred to the new subject so he can present it together with a copy of this credential, where the *issuer* is the *subject* and the new subject is the *holder;* and/or a RC, which expresses this relationship. If a RC is not used, the copy of the DC should include the

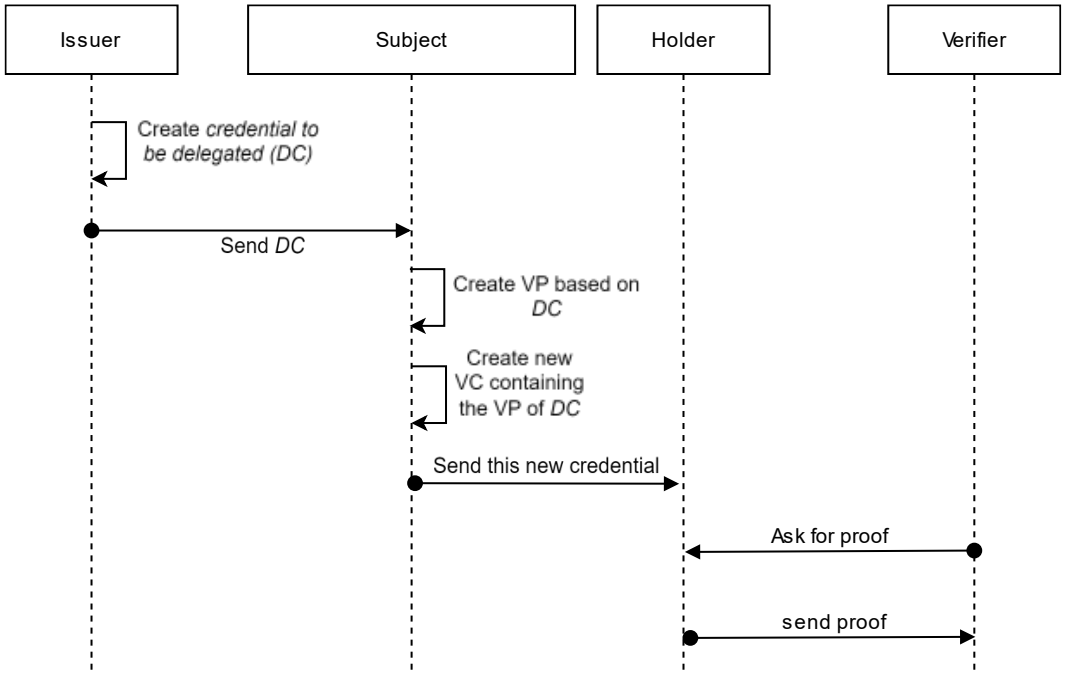

Figure 5-2: *Aries RFC proposal for delegation*

relationship between both in a separated field. The general flow for the W3C proposal has been summed up in Figure 5-1, where the *subject* is delegating a VC to the *holder*. Hence, the *verifier* can ask for a proof directly to the *holder*. Finally, it is worth to mention that W3C proposes the use of their own credentials, known as W3C credentials, which intend to be a standard for VCs.

Aries RFC. Aries proposes, in a Request For Comments (RFC), the creation of a field in a VC containing the VP of the DC created by the *subject*. Consequently, it proposes creating a chain of trust where the different VPs of the different DCs are attached in a field of the VC, called *provenance_proofs*. The *holder* then generates a VP when requested by a *verifier*, that contains the VCs of the *holder*, where some of them will contain in an attribute *provenance_proofs* a VP of a DC, created by the *subject*. The *verifier* can subsequently verify the VP generated by the *holder*, and the VPs included in the VC and therefore accept or reject them if all these proofs are either valid or not. The complete flow of the Aries RFC adapted to the proposed industrial use case is shown in Figure 5-2.

5.2.1 Comparative analysis

Regarding privacy, both the W3C and the Aries RFC have their privacy drawbacks. There is a risk of privacy loss when a holder stores another-subject credential, as the W3C proposes. This has also been noted from the research community. For example, some

authors [82] recognized the possibility of a loss of control when the *subject* delegates a credential to another *holder*; therefore, an authentication mechanism based on public key cryptography is proposed for VCs. However, the verification process becomes complicated because VCs need to be ciphered, and the authors [82] also recognize some drawbacks with this design, *e.g.*, this design is not fully compatible with selective disclosure in VCs. There is also a privacy loss in the Aries RFC because the VP is included in the credential, so the *verifier* has access to all the fields through the VP, unless it is hidden using ZKPs. However, we consider that it is a better alternative in terms of privacy because the credential is still under the control of his original holder and some attributes may be hidden through the use of a pre-generated ZKP or using an object containing an enveloped VC within the VP, as specified by the W3C. This method expresses a secured VC using an enveloping security scheme, such as Securing Verifiable Credentials using JOSE and COSE [24].

The W3C approach might also lead to certain security risks when implemented. In this approach, the *verifier* infers that a group of credentials are related when they are presented together. Therefore, *devices* could steal other credentials from the *manufacturer* and present them with the same RC, which would ultimately make the *verifier* accept this credential as valid. In other words, without verifying the original credential against the *manufacturer*, the sole existence of a RC does not guarantee full security. Aries RFC does not have this risk because the *holder* only presents one credential containing all the information. However, this design comes with its own drawbacks. By including VPs into VCs, their size is increased, leading into credentials with a large size.

Focusing on specific technologies, both Veramo and Hyperledger Indy/Aries, as stated in Chapter 2, are the most advanced technologies for implementing SSI solutions. The implementation of delegated credentials following the W3C standard should be straightforward because all technologies should have support for this type of credentials. However, this is not always true, as some technologies have their own way to implement the standard and some operations, although they are standard-compliant, might not be supported by specific technologies. To validate this assertion, we have carried out some implementation analysis using some technologies, and the conclusions are as follows.

5.2.2 SSI technologies and its support to delegation

Hyperledger Indy and Hyperledger Aries. A first consideration is related to the Indy wallet. Indy agents do not support storing VCs in the wallet if the subject is different from the holder of the credential. This is a technical limitation of Indy-based credentials, which assume that the holder and the subject are always the same person. Indy-based credentials do not have a subject field, because they are linked to their holder, who is also the subject. Consequently, it is not possible for a delegated holder to generate VPs based on these credentials and, thus, we can deduce that Indy is not compatible with the W3C standard. Regarding Aries wallet, specifically when Indy credentials are used, it simply inherits all

the indicated limitations from Indy. However, Aries has added support for W3C credentials, which allows to create DCs following the W3C standard.

A second consideration is related to the verification process when the W3C approach is used in Aries. First, it is necessary to implement an additional verification logic to tell the *verifier* that he needs to iterate over the original credentials and check the existence of another credential, which represent the relationship between the holder and the subject, before discarding a VP containing a VC whose subject is different from his holder. It clearly adds extra complexity when it comes to implementing this protocol. Consequently, it is necessary to make considerable changes within the Indy/Aries code to support this functionality.

Regarding the Aries RFC, Indy currently supports this approach because there is no need to store credentials whose holder is different from the subject (and same happens in Aries). Thus, it is compatible with Indy-based credentials. However, Aries must be adapted to support the Aries RFC because his verification method uses sequence numbers to link the proof request with the rest of the verification process, which does not allow to send the VP, store it and verify it later. This can be indeed useful to prevent from replay attacks. By linking the proof request with the rest of the verification process we avoid that a malicious user can reuse the same proof request to obtain the VP once and again. This mechanism should be implemented by other technologies to prevent this kind of scenarios.

Veramo. Veramo does not present some of the issues that Indy and Aries have. In particular, in Veramo, it is possible that devices store any VC as there are no limitations when it comes to storing credentials whose subject is different from the holder. This is because Veramo natively works with W3C credentials. This adds more flexibility to Veramo against Indy and Aries. However, regarding the verification process, it is also necessary to implement the same logic in Veramo to properly implement the delegation flow, so some changes need to be made in the source code to do that. However, Veramo is still in an early stage of development.

Regarding the Aries RFC, it is currently implementable in Veramo, as well as in Hyperledger Indy. Furthermore, Veramo does not implement any sequence numbers between verification steps as Aries does. This can lead to replay attacks as discussed before

Other technologies. Other technologies, such as those described in Chapter 2, have the same issues. Veres One is still in an early stage of development and the documentation regarding implementation guidelines for VCs is very limited. Jolocom presents a higher maturity level than Veres One, but it still needs to adapt the verification method to support the W3C standard. Regarding the Aries RFC, it is implementable by Jolocom.

Table 5-1 maps the current technologies and their support, without the necessity of changes, with the W3C standard and the Aries RFC approach regarding delegated

credentials. As shown, Aries RFC is supported by most technologies. The overall maturity was determined based on the documentation available for these technologies.

Table 5-1: *Mapping between SSI technologies and their support to the W3C (W) and Aries RFC (R) without changes*

	Support other-subject VCs	**Support verification method**	**Overall maturity**
Indy	R	R	High
Aries	W R		High
Veramo	W R	R	Medium
Veres one			Low
Jolocom	W R	R	Medium

5.3. The need for Collaborative Credentials

Let us consider a "collaborative group scenario" in which a *verifier* should obtain the accreditation of certain capabilities from a group of members to provide a service to that group. The members of the group are not known in advance, and they can even change while the service is provided. In addition, the accreditation of capabilities is expected to be done periodically, so that the service provider is guaranteed that the group maintains their capabilities active over time.

The SSI emerges as a powerful solution that allows individuals to justify their capabilities by using VCs because it provides a trust framework around VCs that allows a *verifier* to trust the information (capability) provided by a holder. When these credentials are created by means of a collaboration of a group, they become collaborative. Note that this scenario makes it impossible to consider the use of standard VCs, even in the case that they are issued to several subjects instead of only one, because the issuance of the credential is a process in which the credential cannot be modified after being issued, thus not allowing dynamic combination of collaborators.

A first solution to this scenario could be for the *verifier* to periodically follow a certain procedure to get the identities of the members of the group, then poll every member of the group for credentials of its capabilities, and finally combine the obtained credentials to check if the requirements are met. This approach not only impacts on the performance (time for polling all members is needed), but also requires that the *verifier* knows who all the *group members* are in advance, thus requiring the implementation of specific discovery mechanisms for every *verifier*, and impacting on group member´s privacy (just consider that the *verifier* only needs a set of capabilities and there is no necessity to know how many and which specific devices provide them). In that sense, CCs enable the development of a

software framework that facilitates the implementation for any *verifier*. Furthermore, real time will be required in some scenarios, which is difficult to achieve without a collaboration protocol. This approach also presents race conditions. If many consumers are competing for a limited number of resources, it is not possible to assign a set of resources to a single group. The *verifier* would not be able to block a set of resources for the whole group. These drawbacks are aggravated when it comes to the IoT, as devices are usually made by different manufacturers, use different protocols and are not always easily reachable, then overcomplicating the logic of the verifier. There is also a scalability problem, because if the number of devices increases, so does the number of communications with the *verifier*.

An alternative solution to the polling strategy is having a member of the group designated as *coordinator*, with the tasks of *i)* acting as a single point of contact with the *verifier* and *ii)* holding a specific type of VC that accredits all the capabilities of the group. First, having a single point of contact considerably simplifies the implementation of the *verifier*, which will need minor modifications with respect to the normal functioning in the standard SSI flow. In addition, the *coordinator* will oversee the management of the group, thus allowing to hide details of the group to the *verifier* to protect members' privacy according to defined privacy policies. The *coordinator* can also use selective disclosure mechanisms to hide sensitive information about the group. Note that some use cases will require that some requirements are met without even knowing who meets these requirements, but the group. Hence, the *coordinator* can act as a privacy proxy for the group and represents an endpoint that anyone can reach to verify some information from the whole group, rather than reaching group members separately. Second, note that this solution heavily relies on the existence of a credential that accredits the capabilities of the group. Due to the dynamic nature of the group in our scenario (members can change over time), this credential cannot be statically issued on the group creation instant. For this reason, there is a need to define a type of credential that fits this scenario. These are the Collaborative Credentials. Finally, the *coordinator* has all the information to ask for a set of resources and block them for the whole group, thus avoiding race conditions.

Taking this rationale into account, we are ready to define in what follows the concept of Collaborative Credential, possible use-cases for these credentials and formally model their lifecycle.

Definition. A CC is defined as a group of VCs generated by means of the collaboration of a group of participants.

This group of credentials might be presented to a *verifier*, thus allowing the group to prove certain capabilities, which ultimately might be useful for certain applications like access control or assertion of capabilities. CCs are compiled only under participants´ authorization, which could be granted in an initial step only, or periodically, thus allowing the verification of validity of a CC over time. CCs also inherit all the characteristics of

VCs. First, they are verifiable. This implies that the VCs that are included in a CC must be signed either by the participants in the group or by other external issuers. In addition, CCs must be revocable and support selective disclosure.

Finally, we have checked if CCs comply with the ten SSI principles. As CCs are composed of a set of VCs, they are congruent and aligned with the 10 principles that characterize the SSI, which are: 1) existence, 2) control, 3) access, 4) transparency, 5) persistence, 6) portability, 7) interoperability, 8) minimal consent, 9) limited disclosure and 10) universal inclusion. VCs meet these principles individually, so CCs also meet them because they are composed by a set of different types of VCs. However, as it will be tackled later in the document, the *coordinator* is a semi-trusted authority may raise concerns on the control and access principles. Finally, the protocol has been defined in such a way that ensures consent, because as it will be discussed later, participants express consent to participate by issuing VCs.

5.3.1. Applications of Collaborative Credentials to the IoTaaS

Let us illustrate the use of CCs for the IoTaaS scenario by extending the first example of Chapter 3. In this example, a person P was a researcher who wanted to use the information provided for some sensors located in a coastline gathering information about tides. Without CCs, the use case had a 1-1 relationship between P and the sensor. However, thanks to CCs, it is possible for P to get information from a group of sensors instead of one (1-n relationship). This scenario is interesting when he wants to combine the information provided by different sensors and they are not independent. For example, P may need an extra of accuracy and not relay on the information provided by a single sensor, so he would need more sensors to combine the information provided by all of them and extract better conclusions.

Furthermore, complicating more the previous scenario, P may require collaboration from the different sensors, e.g., that the tide sensor does not separate much from the wind sensor, otherwise, metrics would not be accurate and therefore could not be correlated as they don´t cover the same area. The tide sensor could be moved very far from the other sensor; in that case, the wind sensor may need information from another tide sensor, much closer to it. In other words, the group should be dissolved, and a new group should be created dynamically, with a new tide sensor substituting the first one. At the same time, this should be transparent to the emergency centre, who does not care about how many or which sensors are providing the information. This simple scenario can be even more complicated by involving other actors, such as a windmill operator (human actor). Imagine that the operator needs to do maintenance works in the wind sensor each 100 days, otherwise metrics could lose accuracy. The emergency centre could ask him to prove that he has done the maintenance on time. This information can be obtained from an intelligent

timer with internet connectivity installed in the windmill that is updated when the operator does the maintenance.

As the reader may have noticed, there is an actual collaboration between different intelligent devices (wind sensor, tide sensor, intelligent timer, etc.) to gather all the required information by the emergency centre, and the group is dynamically created depending on the verifier´s requirements. This cannot be solved with the standard flow proposed in Chapter 4, as it requires to dynamically create groups of sensors and the collaboration of the different members of the group. Furthermore, semi-real time is also a requirement because the group would need to provide this information to the *verifier* periodically with certain time frames, which is especially critical in the proposed use cases.

5.3.2. Other applications for Collaborative Credentials

Focusing on the healthcare sector, imagine a futuristic operating room with an intelligent scalpel. The scalpel will only be unlocked, thus allowing the surgeon to operate with it if some conditions are met. For example, due to hygiene requirements, the surgeon may need to wear a face mask and surgical cap to unlock the scalpel. Consequently, the surgeon will need to prove against the scalpel that he is wearing a face mask and a surgical cap. Furthermore, not every face mask is valid; the face mask must be FFP2. This certification is provided by a trusted certification entity.

This simple scenario can also be even more complicated. Imagine that the surgeon has been working for more than 24 hours, then not being physical and mentally prepared to do an operation. The scalpel could ask the surgeon to prove that he has not been working for more than 24 hours before letting him to do the operation, which can be obtained from an intelligent check-in machine.

Finally, as any machine is prone to fail, or as any extremely urgent circumstance may require, it is necessary to implement a mechanism in the scalpel that allows the surgeon to override any configuration and manually unlock it. However, this should be registered as a manual override to enable further inspections. In other words, it must guarantee non-repudiation over the surgeon actions. Figure 5-3 draws this use case.

As another example, let us consider an implementation in a ski station. An intelligent chairlift is configured to not allow people to get in without a pass, which can be modelled by using a VC. This can be verified when the skier accesses the chairlift. However, the chairlift may also require that the skier is wearing proper boots and certified sunglasses. In other words, the pass will be dynamically generated depending on the equipment the skier is wearing. A collaboration between different elements is indeed required, hence requiring

collaborative credentials. Notice here that not only is the *verifier* checking the state of the equipment but also a *certification entity* can also certify some attributes from it. Consequently, the *verifier* (chairlift) could verify this when the skier is getting in. Furthermore, the chairlift could be interested in doing this verification, for example, each minute, while the skier is ascending. If the verification fails, the chairlift could also emit a notification to the skier kindly asking him to put his equipment on.

Collaborative credentials can also prevent from accidents in the aviation and automotive sector. For example, an intelligent plane might only allow the pilot to start the engine if all the systems have been properly checked. The different elements of the plane, such as brakes or wing sensors, state of the wheels, etc., can collaborate each other and issue VCs to the plane Electronic Control Unit (ECU) certifying that they are properly configured. Only in case all parts are working properly the engine can be started. Furthermore, periodic verifications can be done during the flight to ensure everything is working properly, specially before landing or taking off. Note that the existence of CCs in this scenario simplifies the implementation of the *verifier*, as it does not need to know the exact plane elements, only that they fulfil certain requirements. Normally, critical systems have redundancy, where several components do the same function so there is always one component working in the case that the other fails. Collaborative credentials allow to ensure that, for example, at least one component is up and running. The *verifier* may not be interested in knowing which component is working (in a set of redundant components). Notice that this cannot simply be achieved with a *verifier* doing a polling to verify constantly a set of VCs, because he would need to know all components in advance, which is not realistic due to redundancy and changing environments. Collaborative credentials offer much more flexibility to implement this scenario.

Similar strategies can be applied to the automotive sector. This use case opens the door to another interesting use case for collaborative credentials, which is to prevent

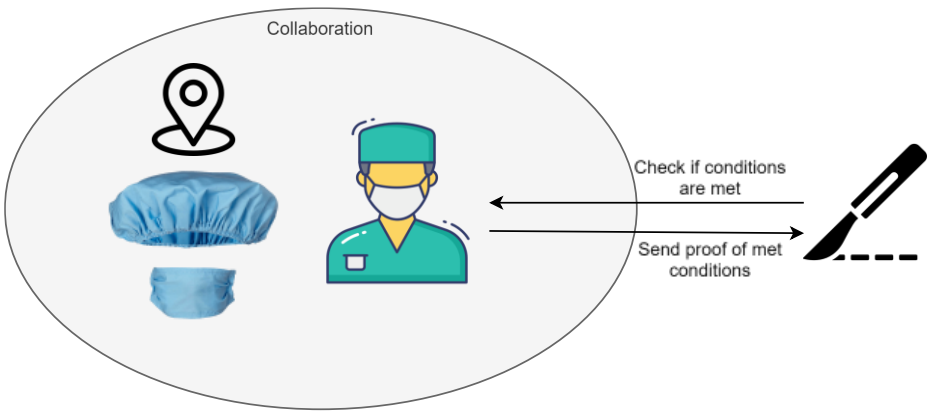

Figure 5-3: *Example of collaboration in the healthcare sector*

impersonation attacks. Each plane component could have a unique identifier and consequent quality certifications. The ECU could ask for a proof showing this identifier and these certifications before letting the engine start. This can prevent, for example, that an attacker physically tampers some components and replace them by tailor-made ones that can compromise the plane security. A fake GPS system could periodically send the attacker the location of the plane, information that the attacker could use maliciously. By checking, for example, the authenticity of the GPS system, we can prevent this scenario. This approach is not new at all, even companies such as Apple have already used it. For example, some users have reported that the TouchID stopped working after they replaced the iPhone glass in a non-official store.

The uses for collaborative credentials are broad enough to be detailed in this document, but in general, they can fit in any scenario that requires collaboration between parties to jointly satisfy a verifier´s requirements in real time.

5.4. Formal model for Collaborative Credentials

This section formalizes the concept of CCs, providing a generic structure for them and presenting the different actors involved in their generation and use.

As stated above, a CC is composed of a set of VCs that are provided by the participants of a collaboration group. Therefore, CC={VC_1, VC_2, …, VC_n}. CCs may be stored in a wallet as a group of individual VCs. Following the W3C [24] specification, typically a VC has the following format:

- Metadata (M): information about the issuer, validity, etc.
- Claims (C): information about the subject.
- Proofs (P): cryptographic information that makes the credential verifiable.

Therefore, considering that $VC = [\mathrm{M}, \mathrm{C}, \mathrm{P}]$, a CC will be defined as: $\mathrm{CC} = \{[M_1, C_1, P_1], [M_2, C_2, P_2], \dots, [M_n, C_n, P_n]\} = \{\bigcup_{i=1}^{n} [M_i, C_i, P_i]\}$, with n being the total number of VCs and $\bigcup_{i=1}^{n} x_i$ the concatenation of n elements x_i.

These VCs can be of the following types:

1. Verifiable credentials signed and issued by the participants of the group themselves (thus acting as issuers) certifying claims to other participants. In the proposed use case, an example of this would be a credential issued by one sensor to another sensor.
2. Verifiable credentials signed by several participants (with a multi-signature process) in which internal claims, i.e., claims that have been generated by these participants, are inserted. In the proposed use case, an example of this would be a credential multi-signed by different sensors, sent to another sensor, containing claims from all

participants. They could be formulated as $CC = \{[\cup_{i=1}^{n} M_i, \cup_{i=1}^{n} C_i, \cup_{i=1}^{n} P_i]\}$ for n participants.

3. Verifiable credentials signed by external issuers with claims referring to participants of the group. An example of this would be a credential issued by a trusted certification entity to a tide sensor, for example certifying that it is IP68 waterproof.
4. Delegated credentials signed by the original holder and sent to another participant. These credentials also include an authorization to be used within their claims.
5. Collaborative credentials. Note that a CC could in turn be part of another CC in an inter-group collaboration. An example of this would be a VC of type 1 or 2 containing another one of type 3. In that case, a CC composed by *k* CCs could be generally formulated as CC={CC_1, CC_2, …, CC_k}

Regarding the composition of VPs containing CCs, the W3C standard [5] supports including various VCs into a single VP, which can also be used to generate VPs based on a CC. Similarly, a VP is composed by:

- Metadata (M)
- Array of *k* Verifiable Credentials ((VC[k]))
- Proof (P)

Therefore, a $\mathrm{VP} = [\mathrm{M}, \mathrm{VC}[k], \mathrm{P}]$.

5.4.1 Actors in the model

The Table 5-2 summarizes the SSI roles that can be played by the different actors, which are:

Table 5-2: *Roles and actors for CCs*

Actors	**Roles**
Certification entity	Issuer
Group member	Holder, Issuer
Coordinator	Holder, Delegate, Verifier
Verifier	Verifier

Certification entity

A *certification entity* is a (optionally qualified) person or legal entity that certifies something about the *group members* by issuing them a VC. In this case, the *certification entity* actor in CCs corresponds with the role of issuer in SSI.

Group members or participants

A CC will be created by a set of at least two members that collaborate with that purpose. This set will be known as *group members* or *participants*. Note that the system may work with only one participant, but then, there would not be any collaboration. The *group members* store their identity and credentials in their own wallet in a self-sovereign way, *i.e.*, they act as SSI holders. In addition, *Group members* delegate the use of their VCs to the *coordinator* or authorize him to present them to prove that this *group member* belongs to the group.

Furthermore, *group members* must be able to support the logic of CCs. They must interact with each other and collaborate to meet a set of requirements (usually specified by a *verifier*). To enable interoperability in the use of CCs, it would be necessary that these requirements follow a specific specification, as it will be explained below in this section.

Coordinator

Among the participants, one will play the role of *coordinator*. The *coordinator* is responsible for receiving the delegated VCs from *group members* and compiling them as a CC. Therefore, the *coordinator* would gather information from the *group members* and then prepare a joint VP and send it to the *verifier*, thus acting as holder from an SSI perspective. In addition, the *coordinator* can oversee the negotiation to let the *group members* achieve the necessary consensus for the generation of the CC. The *coordinator* must be chosen following a certain algorithm, and the specific *group member* playing this role might change over time.

Additionally, the *coordinator* may also act as an SSI verifier, by verifying the information provided by the *group members*, such as, for example, that the signature is correct, or verifying any other information that *group members* may possess to fulfil requirements of the use case. The *coordinator* will be an SSI holder himself but may also act as a delegate for the *group members*. Delegation eliminates the communications between *group members* and the verifier.

Verifier

The *verifier*, following the SSI nomenclature, is the person or entity that verifies some claims from a group of members. In this case, the *verifier* actor in CCs corresponds with the role of verifier in SSI.

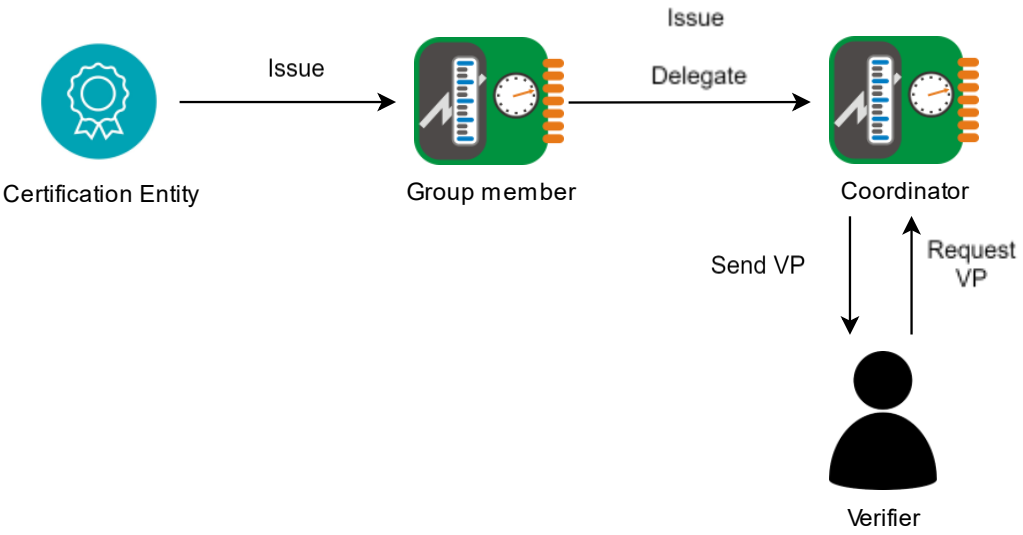

Figure 5-4: *Actors in the model for CCs and their relationships*

To better understand the involved actors, in our use case example, *group members* would be the wind sensor, tide sensor, intelligent timer and the operator; the *verifier* would be the emergency centre; and the *coordinator* could be any *group member*, depending on the chosen consensus mechanism. For simplicity, we will suppose that the *coordinator* is the operator. In our use case example, the *certification entity* could be any entity issuing credentials to the *group members*. For example, a certification agency could certify that a tide sensor is IP68 certified, and a university might have certified the technical degree to the operator. Figure 5-4 represents the different actors and their relationships.

5.4.2. Identification of actors

Each *group member* must have its own DID to be identified and its own private and public keys stored in its wallet for signing VCs. These DIDs allow the identification *group members*. DIDs and their correspondent DID Document creation and registration in the Verifiable Data Registry (VDR) are done similarly as proposed in Chapter 4.

Registering DIDs in a public ledger can be problematic when the number of devices is high, as usually happens with the IoT. Hence, private DIDs solve this problem as they do not require to be registered in the ledger. However, some technologies, such as Hyperledger Aries, require that those actors that issue credentials register their DIDs and DID Documents in the ledger. Therefore, a private approach for the ledger, or limiting the number of devices could be preferable if Hyperledger Aries (or another technology with a private DLT) is used for the implementation. However, having a private ledger is not exempt from limitations. If the ledger is kept private, it is not possible to make this solution

global. Another option is to allow different IoT devices to have the same DID. For example, focusing on the use case example, it has no sense to have thousands of DIDs for thousands of sensors, being more appropriate that all sensors in a batch share the same DID. This DID should be registered in advance, for example when a batch of sensors arrives to the warehouse. Furthermore, each actor must have a public endpoint where it can be reached by resolving its DID. This alternative seems to be more practical for our use case, but different use cases may opt for another option. Therefore, in our approach, as all *group members* can issue credentials, we propose that all DIDs are registered in the VDR at the beginning.

Similarly, the *certification entities* will do the same process of generating their own DIDs. A *certification entity* DID, together with its corresponding DID Document, will be stored in the VDR after their creation. Again, this would be problematic if a private ledger is used because external *certification entities* would not be able to reach the ledger to do the registration. In this case, *verifiers* could query information in a hybrid way in a private ledger and also in a public ledger, where certification entities can register their DIDs.

5.4.3. Conditions specification

The *verifier* must follow a specific conditions specification (*CS*) for requesting information to the *coordinator*. Consequently, the *CS* is created by the *verifier*, containing his needs, and sent to the *coordinator*. Therefore, the *coordinator* can accurately provide to the *verifier* the information that he needs. In other words, a common *CS* ensures compatibility between the verification applications and the group. IoT devices are often heterogenous (as will be discussed later in the document), and made by different manufacturers, so the CS helps to manage this heterogeneity by specifying the way in which these devices need to communicate to become part of a group. Furthermore, *group members* must also know the *CS* to provide the *coordinator* the information using the format it is expecting and to decide if they join the group or not according to the collaboration purpose. Actors must agree on the *CS* format and structure in advance; hence it can be specified during the onboarding process, or it could be published elsewhere and be referenced through a URL during the onboarding. The best option if the number of devices is high, as usually happens in the IoT, is to publish it in a public web or directory, as is usually done with the VC schema contained in the *@context* field of the credential, following the W3C guidelines; and then reference this URL during the onboarding. Furthermore, all actors must share or be aware of a common context that allows to understand this *CS*, because some variables, such as current time, may require a common context. This context can also be included in the public directory or shared during the onboarding.

First, the *CS* must include list of key-value attributes, each one representing a capability of the group, as requested by the *verifier*: {$Attr_1$: $value_1$, $Attr_2$:

$value_2$, …, $Attr_n$: $value_n$}. Some of these attributes will be delegated to the *coordinator* from the *group members*. G*roup members* might also be able to generate ZKPs based on these attributes so they can proof they have certain attributes without disclosing the exact value of these attributes. A ZKP applied over an attribute generates a predicate. Hence, the operator may use a ZKP, for example, to generate a proof that he has done the maintenance using the *maintenance_time* attribute. Section 5.7 will deep into this example. More information about how ZKPs work can be read at [28]. The `count` label indicates the number of *group members* of this type that the verifier is requesting. Attributes and predicates are labelled under `gm_type`. This label indicates the type of *group member* that the *verifier* is needing. For example, a verifier could need information from a tide sensor, wind sensor, operator, and an intelligent timer. This will be exemplified in Section 5.7.

In addition, the *CS* should include the purpose, which at least should include the collaboration reason and the duration of the collaboration and certain requirements for participants to be eligible for the group. This collaboration reason must be understood by all network participants and should allow *group members* to see if they meet the requirements for their participation in the group or not. That is, *group members* may be configured not to participate in groups with a certain purpose, or with other *participants* who do not meet certain characteristics. This content of this field will depend entirely on the use case and should be defined at implementation level.

While people have the intelligence to take decisions, when it comes to IoT devices, this logic must be programmed by someone else. Requirements are different from attributes and predicates: they represent everything that is required to participate in the group but that is not included in a VC, and therefore is not verifiable. In other words, the verifier cannot trust that the *coordinator* is complying with a specific requirement. This is not a limitation, because if the verifier wants to be sure that one or several *group members* meet a specific requirement, he would only need to include this into the components section, so *group members* would need to generate a VP proving that they meet this requirement, certified by a trusted certification entity. There is a trade-off between security and usability, because if the verifier requires too much information certified by a trusted certification entity, there will be less participants that can get this information, and therefore it would be more difficult for the *coordinator* to create a group that fulfil these requirements.

Typically, requirements are used to specify hardware and software requirements (for example, minimum RAM or CPU). Within the requirements, we could differentiate between requirements for all the participants and for specific participants. All the fields in the *CS* should follow a specific ontology, so all *group members* understand the same message when receive the CS. This *CS* depends on the use case and should be known in advance by all actors.

To sum up, a *CS* can be generally formulated as shown in Figure 5-5:

```
{
  components:
  {
      gm_type1:
      {
                  {
                  Attr1: value1,
                  Attr2: value2,
                  …,
                  Attrn: valuen,
                  count=value
                  }
      },
      …,
      gm_typek:
      {
         {…}
      },
  }
  purpose:
  {
    reason: “”,
    duration: “”,
    …
  },
  requirements:
  {
      all:{Req1, Req2,…, Reqm},
      gm_type1:{ Req1, Req2,…, Reqm },
      …,
      gm_typek:{ Req1, Req2,…, Reqm }
  }
}
```

Figure 5-5: *General formulation for a CC*

5.4.4. Lifetime and revocation

An interesting aspect of collaborative credentials is that, since they are a composition of credentials contributed by the participants of the group, their use must be subject to the authorization of the participant group. That is, when such authorization ceases to be valid, the credential must also be invalidated (automatic revocation mechanism). This would potentially apply to use case scenarios where real-time restrictions apply. For example, participants could leave the group if others do not meet specific security requirements.

5.5. The Collaborative Credentials´ lifecycle

In this section, the complete collaborative credentials' lifecycle is presented, together with its consequent phases. This lifecycle has is summarized in Figure 5-6:

5.5.1. Initialization

Initially, as explained, all actors must create their own DIDs for identification. These DIDs can be either private or public. Please, refer to section 5.4.2 for some considerations about creating public and private DIDs for actors in a CCs framework.

5.5.2. Presentation request phase

Once the initialization has taken place, a device will initially request a service to the *verifier*, and this operation will automatically make that device to assume the role of initial coordinator of the group, who will act as an initial endpoint. The *verifier* will send a presentation request to the initial *coordinator*. This presentation request contains the *CS*. The opposite situation can occur. The verifier could request a service provided by a group of devices, so the *verifier* will request the service to a device (then becoming the initial *coordinator*), who will answer the request, and the *verifier* would then answer with a presentation request.

Once the initial *coordinator* receives the presentation request with the *CS*, it needs to create the group. Thus, the collaboration phase starts.

5.5.3. Collaboration phase

Discovery

The proposed model requires that potential participants in the group are first identified before any communication can even take place. It is therefore required to define a discovery protocol, at least once, together with the onboarding and coordinator selection processes. Ideally, once it has been defined, there will be not necessary to repeat them in every iteration. The discovery might vary entirely depending on the use case.

One alternative is using a marketplace, as proposed in the IoTaaS business model in Chapter 3. *Group members* can be listed in one or several marketplaces and found by the *initial coordinator.* The use of a marketplace is not the unique alternative.

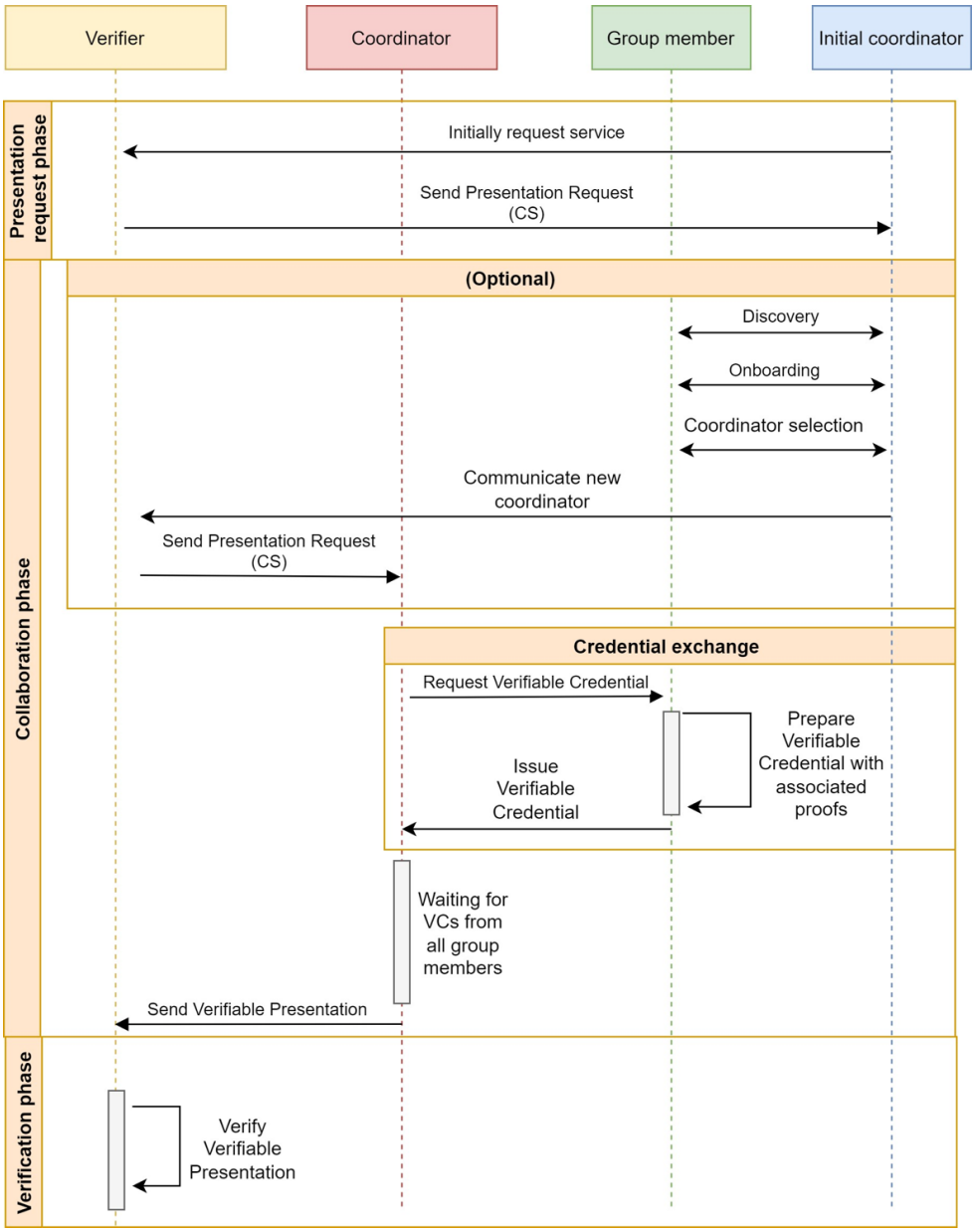

Figure 5-6: *Different phases and steps in the CCs lifecycle*

Group members and *coordinator* might be part of a shared network medium, and the *coordinator* might reach them with a broadcast message. An alternative approach involves using multicast addresses to create groups of devices or follow a publish/subscribe communication model to reach devices interested in joining to groups. This seems to be an

interesting approach for industrial scenarios, where all the actors are in an isolated network, rather than connected through the Internet.

In general, a discovery protocol would allow the initial *coordinator* to discover and identify candidate *group members*.

Onboarding

Once the discovery has finished, it is required to implement another protocol, termed Onboarding. This protocol is used to establish a one-to-one communication between the *group members*. Different topologies can be established for communications in the onboarding. In general, it could be a communication in a tree structure with the *coordinator* as the root, but it is possible for other scenarios to have a full mesh topology. In fact, a full mesh would add robustness and fault tolerance in scenarios where the *coordinator* is expected to change with certain frequency, as communications can continue without a new onboarding phase.

There are many ways to implement this protocol. For example, some use cases will simply require a TCP connection, which is the standard onboarding process in client-server architectures. Some protocols, such as DLTS (Datagram Transport Layer Security), which works over CoAP (Constraint Application Protocol), a common protocol for the IoT, use a 3-way handshake [83] between clients and servers. DIDComm [60] can also be used to implement an onboarding protocol by using a DID exchange between actors, thus providing an extra layer of security (confidentiality and authentication). Note that SSI relies on DIDComm as the standard protocol to onboard users.

The onboarding protocol would include a set of onboarding messages to reach the goal of establishing a communications channel. These onboarding messages must somehow include the CS or a URL referencing it, so that onboarding candidates can decide if they join the group or not. For example, *group members* could check if their credentials have the required attributes for the collaboration before joining the group; or they could interrogate other *group members* or the *coordinator* to check if they meet some basic security requirements before joining the group.

In general, the decision to join or not the group will entirely depend on the use case, as sometimes *group members* might be forced to join it, managed by the *coordinator*, so that *group members* could act as slaves, following the master´s orders. Furthermore, deciding when to end the collaboration is also entirely dependent on the use case. Some uses cases will require that actors can end the relationship unilaterally, while others will require consensus to end the collaboration.

When the onboarding process ends, actors have been onboarded and therefore they have set up a temporal or permanent communication channel.

Optionally, once the onboarding has finished, the *coordinator* can build a table that relates each of the requirements requested by the *verifier* to the different *group members*, so that it knows which *group member* must provide which part of the requirements for the CC compilation. This is indeed useful to replace a *group member* that is no longer available by another one that meets these requirements. This table can be named GMRT (Group Member Requirements Table). Another interesting discussion is if this table must be locally managed by the *coordinator* or shared among the *group members*. By sharing this table among the *group members*, the solution becomes fault tolerant, because they all have the information needed to recreate the group without the participation of the *coordinator*. The group is formed after the onboarding process after consent from the *group members*.

Coordinator selection

Recall that the initial *coordinator* is the participant that started the process and thus, originally receives the presentation request. Yet, in a scenario where the *verifier* expects periodic validations of CC (let us call *iteration* to every periodic validation), there are some circumstances where it might be desirable to change the participant that act as *coordinator*. For example, the *verifier* might request some requirements in the *CS* for the *coordinator* that the initial *coordinator* is not able to fulfil, so that *group members* decide to choose a more capable *coordinator*. In general, when this change of *coordinator* is needed, a coordinator selection phase should be executed.

Several strategies can be followed for electing a new *coordinator*. For example, *group members* might vote to elect a *coordinator*, or it could be randomly chosen. Decentralized voting systems [84][85] could be used to elect the *coordinator* amongst the *group members*. *Group members* could vote to decide whether the *coordinator* must be replaced or not. How frequently the *coordinator* is re-elected is another aspect to consider at implementation level. Additionally, different criteria might be considered to trigger the coordinator selection, as geographical proximity to the verifier or computational power of the candidates. In general, the selection of the *coordinator* will entirely depend on the final use case.

After selecting the new *coordinator*, the initial *coordinator* communicates the result to the *verifier*, so that the *verifier* forwards the presentation request to the new *coordinator*. The *verifier* would need to contact with the new *coordinator* and start again the process, therefore implying a delay to complete the current iteration. Depending on the use case, this could be optimized. For example, the initial *coordinator* could communicate with the *verifier* forever. This has the advantage that the group management is transparent to the *verifier* as he will always communicate with the same actor. However, it would not be valid in the case that the initial *coordinator* is not capable of assuming his role, for example, due to insufficient computational power, so it does not fit every use case.

There is also the possibility of publishing the *coordinator* in a public directory, so the *verifier* can query this information. This might not have much sense if there is just one *verifier*, but it could be considered with more than one *verifier*. Deciding between one option or another will depend on the requirements of the use case.

The discovery, onboarding and coordinator selection are necessary in the first presentation request, but in subsequent requests would not be carried out since the group is already formed (no discovery or onboarding is needed) and the *coordinator* has been chosen for a period.

Credential exchange phase

In this phase, the *coordinator* requests for VCs to the *group members* according to the *CS*. Then, *group members* will send their VCs to the *coordinator*. Once the *coordinator* has received all the VCs from the *group members*, it creates a VP with all the required information and send it to the *verifier* as a response to the presentation request sent in the first step.

If a predicate over an attribute is requested in the *CS* instead of the value of that attribute, the *coordinator* could generate a ZKP over the corresponding attribute.

Each participant contributes to the group by sending a VC to the *coordinator* certifying its capabilities. The *coordinator* does not control the DIDs of the *group members* (is not a DID controller) nor their private keys.

5.5.4. Verification phase

This is the last phase of the protocol. The *verifier* needs to perform two verifications. First, it needs to verify that the *coordinator* has been authorized by the *group members*, in other words, that it can generate a VP based on the VC provided by the *group member*, and secondly, he needs to verify the delegated credentials included by the corresponding *group member* and check if the CS requirements are really met. If both verifications are correct, the *verifier* can be sure that the *coordinator* fulfils all the requirements.

5.5.5. Revocation

Revocation must be supported in CCs. In general, there are several scenarios in which a CC might be revoked:

- One or several *group members* leave the group, and no other *group members* can replace them;
- The *coordinator* has changed;

- *Group members* or the *coordinator* have been hacked and the security of the system is at risk;

Also, the *certification entities* may revoke their issued credentials. In this case, *group members* should not be able to prove anything with their revoked credentials. Revocation is done following the W3C standard for VCs [24]. *Group members* or the *certification entity* can revoke their issued credentials anytime. By doing so, the CC is automatically invalidated, because it is composed by individual revoked VCs and therefore the *coordinator* cannot prove anything from it.

5.6. Formal security analysis

In this section, some security considerations are presented. The lifecycle presented before can suffer from several attacks or misuses from the involved actors. Some of them are presented here, as well as possible solutions. We have identified the assumptions, attack vectors, variants (if applicable) and possible solutions, similar to those proposed by other authors in protocols such as OAuth 2.0 [87] or OpenID Connect [88]. We have focused on the specific attacks that affect the CCs lifecycle, excluding those affecting the systems of the end users (e.g., misconfigurations, local vulnerabilities, etc) and those affecting the network (e.g., sniffing, Man-in-the-Middle, etc)

Initial assumption: All actors are identified by their DIDs and VCs are signed.

5.6.1. Malicious group members attacks

Attack 1. Group member delays in sending its VC

Assumptions	Group member has already joined the group.
Attack Vector	A *group member* could delay the whole system if it is late in sending its VC to the *coordinator*. As the *coordinator* cannot send the VP to the *verifier* until the whole group has sent him the required information, this situation can compromise real time requirements.
Variant	A similar problem might happen if a *group member* blocks the group creation. However, this can be easily solved in terms of security by finding another member for the group.
Fix	This can be solved by using a timeout mechanism. After a timeout is reached, the *coordinator* should find another member and request it to join the group to get the missing requirements or capabilities, just in the case that there is no other *group member* that can provide them. This, of course, would negatively impact on the performance, as the *coordinator* would need to discover and onboard this new member. Non-cooperant *group members* could also be blacklisted, so re-entrancy (*group member* constantly entering the group and delaying) is avoided.

Attack 2. Group member performs a DoS against the coordinator

Assumptions	Group member has already joined the group.
Attack Vector	A malicious group member executes a denial of service (DoS) by sending a high number of credentials to the coordinator.
Variant	This same problem can occur if the verifier sends many presentation requests in a short period of time.
Fix	To prevent that, the coordinator should be programmed to reject credentials that have not been requested or are sent in a shorter time interval than desired. This avoids saturating the coordinator.

Attack 3. Group member changes the coordinator selection result

Assumptions	None
Attack Vector	When it comes to the coordinator selection phase, a *group member* might try to alter it and influence the election of the *coordinator*.
Variant	The *coordinator* itself may alter the coordinator selection result.
Fix	To prevent that, the chosen consensus algorithm must be byzantine fault tolerant; in other words, it must allow to reach consensus even if some participants respond with incorrect information. This can also prevent that some *group members* confabulate to veto other *group members* or substitute the *coordinator*. However, if the number of confabulated *group members* is big enough, this situation is inevitable. The byzantine formula that expresses the maximum number of colluding participants is $n=3f+1$. That means that, with n participants, only f can be malicious to ensure the system keeps working. In case it is the *coordinator* the one who tries to change the result, this can be avoided by publishing the results of all group members so they can see the recount. Decentralized voting through the use of technologies such as Blockchain is also a solution.

Attack 4. Group member impersonation

Assumptions	*Group member* has already joined the group.
Attack Vector	Although all communications are signed, an attacker might try to obtain the keys and therefore completely impersonate a *group member*.
Variant	None
Fix	This might be prevented using technologies such as TEEs or TPMs. A trusted execution environment (TEE) is known as an isolated processing environment in which applications can run, separated from the rest of the system. This can be used to protect the signing process as well as the data (cryptographic keys) involved in it. The signing process can also be protected by employing embedded cryptographic keys using a trusted platform module (TPM) for embedded systems

5.6.2. Malicious coordinator attacks

The *coordinator* is indeed an entity which can raise privacy concerns as *group members* relay on it to communicate with the *verifier* on their behalf, but it requires an authorization from the *group members* to guarantee that the consent principle is fulfilled. VCs issued by the *group members* to the *coordinator* do not include sensitive information about them, but an authorization to participate in the group, together with a set of capabilities selected by the *group member*. In other words, *group members* choose which information want to share with the *coordinator*, so although this information is under the control of the *coordinator*, the risk is mitigated. Finally, regarding the access principle, *group members* have always access to their own data, and they share part of this information with the *coordinator* when they are requested by sending him a VC. In that sense, access is also fulfilled. Besides, this VC can be revoked at any time by its issuer (*group member*). However, the system can suffer from a sybil attack by a malicious *coordinator*.

Attack 5. Sybil attack

Assumptions	Although the *coordinator* can simulate being a whole group itself, he cannot fake capabilities as they must be signed by a trusted *certification entity*, hence reducing the impact that this attack can make, because at the end, the *coordinator* will need to fulfil all verifier´s requirements, either in group, or either alone. The group has not necessarily been created in advance.
Attack Vector	In a sybil attack, an attacker can generate numerous pseudonymous identities, which appear as distinct entities but are all controlled by the same malicious entity, in this case the *coordinator*. In other words, the *coordinator* can simulate to be backed up by a group but in reality, be alone.
Variant	None
Fix	It can be prevented by using public DIDs with a corresponding DID Document registered in the VDR. However, having thousands of DIDs for thousands of devices stored in a public registry is not always possible, and hence this is a limitation of the framework. In addition, this sybil attack can be eliminated by requiring *group members* and the *coordinator* himself to authenticate with a credential that allows them to join a specific group, issued by a third party. However, this authentication process also involves a time cost. Further research works can explore this area.

Attack 6. Bypass group member´s permission

Assumptions	*Group member* has already joined the group.
Attack Vector	A malicious *coordinator* might create a VP by including an old VC issued by a *group member* without its permission.
Variant	None
Fix	This can be avoided if the *group member* either adds a “validity” field in the VC or revokes it. *Certification entities* should also either properly revoke their credentials or include a “validity” field in them to prevent this problem.

Attack 7. Coordinator performs a DoS against the verifier

Assumptions	None.
Attack Vector	A malicious *coordinator* can delay in sending the requested VP to the *verifier*, therefore making the *verifier* wait indefinitely.
Variant	None
Fix	*Verifier* implements an internal timeout after which, if no VP is received within the specified time, it attempts to form another group with a different *coordinator*.

Attack 8. Initial coordinator communicates incorrect coordinator

Assumptions	*Group member* has already joined the group.
Attack Vector	The initial *coordinator* intentionally communicates a wrong *coordinator* to the *verifier*. The *verifier* then would communicate with an unofficial *coordinator*. This attack vector only affects the *verifier*, because *group members* know who the real *coordinator* is, so they will not accept requests from another participant.
Variant	None.
Fix	There are two ways to fix this: 1) a public directory where the *coordinator* is published, 2) a timeout mechanism in the *verifier*. As *group members* will not trust an unofficial *coordinator*, they will not send him their authorization and therefore the *coordinator* will not have anything to present to the *verifier*. The result therefore is a DoS to the *verifier* and can be solved with a timeout. An alternative fix to this problem is that *group members* issue a VC to the *coordinator* certifying its role.

5.6.3. Final remark

Finally, some particular use cases may imply an additional logic for security reasons. For example, what would happen if a car or a plane cannot be unlocked for any reason and the person needs to drive? In real life, some scenarios require that humans override the machine logic. In this case, the car must have a manual override for emergency situations. As another example, an operator may need to override the verification for interacting with some devices during an emergency. The system must, at least, guarantee non-repudiation, so the driver cannot deny that he has activated the manual override.

5.7. An implementation for Collaborative Credentials

Starting from the abstract model for CCs and its lifecycle presented in previous sections, now we show an implementation example CCs focused on the use case presented previously in this Chapter, detailing the implementation for every aspect defined in the collaborative credentials' lifecycle. Our aim is to show the feasibility of the CCs model by showing the details and performance of a simple implementation. The technology used is Hyperledger Aries for the SSI agents of the different actors and Hyperledger Indy for the VDR, due to its compatibility with delegation, as studied in Section 5.2.

When designing CCs we must take into account the features that characterize the IoT. More information about these features is provided in Section 1.1. One of these features that requires special attention is the heterogeneity. The idea behind the proposed implementation of CCs is that they can be enabled in any device that supports HTTP methods, thanks to the modular design of Hyperledger Aries, which separates the agent (in a cloud server) and the controller (IoT device). IoT devices are commonly made by different manufacturers, occasionally using different protocols, which often complicate their communication. Aries agents are designed to run on servers, being not intended for mobile device deployment. To interact with the agents, developers need to implement a controller that communicates with it by sending HTTP requests and receiving webhook notifications. IoT devices are capable of performing these HTTP requests, because by definition, they need to connect to the Internet, therefore being able to run controllers without any further issues. Due to this modular design, it is adequate to be used in the IoT. By minimizing the piece of code that runs on the IoT (controller), devices with constrained resources are feasible for using controllers, while the SSI logic (agent) runs on a separate constraint-free server. It also allows developers to use any language of convenience for implementing the HTTP requests. In our case, Hyperledger Aries controllers have been implemented using Python language and Flask for web development.

First, it is necessary to implement a delegation scheme, for which we propose the Hyperledger Aries RFC delegation scheme [67]. The delegation has been implemented using the Hyperledger Aries and its source code has been modified to support delegation. In particular, including the delegation functionality has been one of the biggest challenges of the research work. This has also been done by preserving backward compatibility. One of the results of the present thesis has been a modified Aries agent that is able to communicate with other Aries agents and also support delegation. As presented previously in this Chapter, Aries does not support delegation directly, so it is necessary implement it by updating the code.

The chosen delegation scheme has the drawback that it does not allow the delegate (*coordinator*) to generate a ZKP over the delegated attributes in the delegated credential.

Consequently, a limitation in this specific implementation is that the predicates that can be specified in the *CS* are limited to the attributes in the non-delegated credentials. This is a clear drawback of the proposed implementation that other implementations in the future can tackle.

The proposed *CS* for the implementation follows a JSON format and is represented in Figure 5-7. This means that the emergency centre *(verifier)* is interested in a group where an operator is present (operator_state:ON), and he is operating a windmill that contains a wind sensor (wind_sensor_state:ON_operator), and a tide sensor, that at the same time is IP68 (certification:IP68) and it is within the operating range of the wind sensor. The location of the wind_sensor has been internally set up in the tide sensor to (43.41, -2.99) and an internal variable in the tide sensor mocks a variable location. When both locations are separated by less than 1km, the tide_sensor_state becomes "ON_wind_sensor". This distance has been pre-defined, but it could be also variable, in that case it should be included in the *CS*.

In addition, the operator has been doing the maintenance periodically each 100 days ($8x10^6$ seconds), based on the time that the intelligent timer recorded. It would be possible to include simple regular expressions in the *CS*, for example to indicate that the tide sensor needs to be IP68 OR IP67. If not specified, the `count` equals to 1. A predicate can be calculated in the example based on a *maintenance_time* attribute, which contains the time where the operator did the maintenance, by doing current time minus the *maintenance_time* and proof that this value is less than 100 days, using ZKP. That means:

```
current_time- maintenance_time<=8*10^6 (secs)
```

Therefore, `maintenance_time<=current_time – 8*10^6`. The *_operator* in all attributes within the wind_sensor indicates that the operator must operate this sensor and in the case of the intelligent timer, that the operator is the one who has done the maintenance. We have also specified a purpose reason, and a duration of 1 minute (in seconds). That means that the collaboration is valid for 1 minute, and then, the group will be dissolved, and it will be required to create a new group from scratch. Furthermore, all *group members* will need to have at least 0.5GB of RAM memory and must guarantee a maximum delay of 10 seconds in getting a response from the *coordinator* (COO_delay). It is possible to identify individual requirements for *group members* within the requirements label by including the type of the *group member*, as done with the attributes.

CS must be publicly available for all group members, so it should be published in a directory or distributed it at the beginning as this implementation proposes. This *CS* is included in a standard Hyperledger Aries presentation request, encoded in base 64, in the field *comment*. Any other free-text field could be used for this purpose.

```
{
    components:
    {
      tide_sensor:
      {
         tide_sensor_state:ON_wind_sensor,
         tide_sensor_certification:IP68
      },
      wind_sensor:
      {
         wind_sensor_state:ON_operator
      },
      operator:
      {
         operator_state:ON
      },
      intelligent_timer:
      {
        maintenance_time <=
        (current_time – 8*10^6)_operator
      },

    },
    purpose:
    {
        reason: “collaborative cred. PoC”,
        duration: “60” #in seconds
    },
    requirements:
    {
         all:
         {
             RAM: “>=0.5”,  # >=0.5GB of RAM
             COO_delay: ”<= 10” # < = 10 secs
         }
    }
}
```

Figure 5-7: *A CS implementation for CCs*

This presentation request must request the field *provenance_proofs* to support delegation, as recommended in the Aries RFC [67]. The proposed implementation uses a set of three attributes for the new VC, which are: *i) provenanceProofs*: it stores the VP of the delegated credential; *ii) provenanceSchemas*: it stores the credential schema of the delegated credential; *iii) ProvenanceDefinitions*: it stores the credential definition of the delegated credential. The *provenanceProofs* attribute solves the problem of storing a VC with a different subject in the wallet and avoid generating a VP based on this VC, which is not possible as discussed before. The *provenanceSchemas* and *ProvenanceDefinitions* attributes eliminate the necessity of querying the VDR and make the VP self-contained. Additionally, another attribute could be added to control the usage of the delegated credential. For this purpose, a *nonTransferable* attribute with a Boolean true/false value is used to indicate that a credential cannot be delegated. The *nonTransferable* property is

proposed by the W3C and indicates that a VC must only be encapsulated into a VP whose proof was issued by the credential subject. This model can be extended by adding as many attributes as required for more complex scenarios. The labels proposed by Aries RFC are considered to act as separators between different elements and help to retrieve them when required. Thus, we use *[proof]*, *[schema]* and *[definition]* labels to separate them.

Then, the *coordinator* receives both the *CS* and the presentation request, and then knows exactly what information needs to get from each type of *group member*. The *coordinator* will map the *attr_name* within the Aries presentation request with the attribute name in the field "components" of the *CS*. In the example, the *coordinator* retrieves tide_sensor_state, wind_sensor_state, operator, maintenance_time and provenance_proofs from the presentation request and he knows thanks to the *CS* which typology of *group member* may have the information that the *verifier* is requesting. Furthermore, the field tide_sensor_certification is a characteristic of the tide_sensor, not of the *coordinator*, and therefore is delegated. Consequently, its presentation must be included inside the provenance_proofs field requested in the presentation request, as the Aries RFC [67] proposes.

This exemplifies that *CS* can vary depending on the use case and the technology used for the implementation.

Discovery. The proposed discovery protocol will be a broadcast inside the network, where all devices are located. Consequently, the initial *coordinator* will broadcast an UDP message to a specific port where *group members* are listening and will answer if they desire to be reachable.

Onboarding. In a standard SSI procedure, it is required that participants exchange an invitation to proceed with the onboarding. Normally, this is done through a QR code that participants share and scan with their cameras. In this IoT scenario, we propose sending this invitation through the UDP communication channel during the discovery phase.

Furthermore, DIDComm is proposed as the onboarding protocol as it is straightforward to implement it using DIDs as decentralized identifiers. DIDComm is used to onboard *group members* with the *coordinator*. In other words, a tree topology with one root (*coordinator*) has been implemented. Our implemented onboarding message includes the *CS* formatted in JSON and encoded in base64. *Group members* will receive this message. As explained, in a real use case the semantic values used within the CS should be standardized following a certain ontology, as the *group members* will be devices that need to parse this information and compare it against their logic to decide if they accept joining the group or not. In this case, a simple, locally managed GMRT has been implemented in the *coordinator* using a local mapping structure in the code. The *coordinator´*s controller has information about which requirement maps with what *group member*.

Coordinator selection. The implemented *coordinator* selection protocol is a voting protocol. Once an initial *coordinator* has been preselected, it is time to decide who the *coordinator* will be. The selection protocol is distributed, as the different *group members* will execute their algorithm and consequently vote for the *group member* that will become the *coordinator*. In this example implementation, we select a simple protocol in which *group members* choose the participant with the minimum DID. Used DIDs follow the format *did:sov:<identifier>*. Therefore, the identifier is retrieved, and the ASCII code is calculated for each character of the identifier, forming a number with it, concatenating all the numbers, and then retrieving the lowest one. *Group members* receive all DIDs from the *coordinator*, and then calculate the minimum one following this procedure. As DIDs do not change during the experiment, the *coordinator* will always be the same, but it will be chosen based on this algorithm in each round. This algorithm negatively impacts performance and there is no need to change the *coordinator* constantly. Ideally, the coordinator should be chosen based on objective parameters, such as computational power, and change never. However, for the present work, it serves as a worst-case scenario as it will imply recreating the group constantly and will allow to get interesting metrics in Chapter 6.

Once the *coordinator* has been chosen, the initial *coordinator* will send a message to the *verifier* including the DID of the new *coordinator*. The *verifier* will resolve this DID from the VDR, obtaining an endpoint, and will forward him the presentation request. In the case that the new *coordinator* does not have a public DID, but a private one, this endpoint is included in the message instead of being resolved.

Credential exchange. The *coordinator* sends a Hyperledger Aries VC request to each group member asking for its credential/s. Once the *coordinator* gets all the information from the *group members* (issuers/delegators), it will create a VP that match the *CS* in the presentation request sent by the *verifier*. In the example, the *coordinator* generates a VP with VCs that contain the attributes: tide_sensor_state, wind_sensor_state, operator, maintenance_time and provenance_proofs. This last field includes the VP of the delegated attribute tide_sensor_certification. Also, for the case in which a *group member* delays in sending its credentials to the *coordinator*, a timeout of 10 seconds is set up, so that when it is reached the *group member* is expelled from the group, hence requiring a new group creation from scratch in our implementation. The *coordinator* may also need to proof something from himself to the *verifier*. In that case, our proposal is to use self-issued credentials. The *coordinator* will issue a VC to himself. Then it will be able to create a VP.

Verification. The *verifier* then will verify the VP sent by the *coordinator*. This VP will contain a VC with one attribute (*provenance_proofs*) containing an embedded VP of the delegated credential, following the guidelines in the Aries RFC delegation scheme [67].

Once the *verifier* has received the VP, he will verify the different attributes represented in Figure 5-8:

```
Verify
  ({tide_sensor_state,
    wind_sensor_state,
          operator_state,
    maintenance_time<=current_time- 8*10^6
  })
If verified:
    Verify(provenance_proofs)
```

Figure 5-8: *Verification process of the implemented CC*

The VP containing the tide_sensor_*certification* attribute is included in the field *provenance_proofs*.

5.8. Summary of the chapter

This chapter has introduced and explored the novel concept of CCs within the context of SSI and the IoT. CCs address a gap in current SSI standards by enabling groups of entities to collaboratively prove their collective capabilities to a *verifier*. The implementation of CCs builds upon the concept of delegated credentials, extending it to support dynamic group collaboration scenarios. CCs offer significant advantages over traditional approaches in terms of performance, privacy, and scalability, especially in IoT environments.

A formal model for CCs has been proposed, defining the roles, structure, and lifecycle of these credentials. Several potential applications for CCs have been identified across various domains, including healthcare, supply chain management, and IoTaaS scenarios. The security analysis of CCs reveals both strengths and potential vulnerabilities that need to be addressed in practical implementations. Finally, an implementation for CCs has been proposed.

Overall, CCs represent an extension to the SSI paradigm, enabling new use cases and enhancing the flexibility of identity management in collaborative and IoT scenarios. However, further research and real-world testing will be necessary to fully realize their potential and address any implementation challenges.

Chapter 6.

Performance evaluation

This chapter intends to provide empirical evidence that the concepts, protocols and constructions provided along this thesis can be implemented in real life, which corresponds to the objective E) of the thesis (see Section 1.3). Furthermore, it also aims to provide metrics about how intensive are and how much overload the proposed models introduce to existing architectures. A total of four testbeds have been designed to evaluate the models:

1. Test the performance associated to the creation, exchange and verification of the *consumer credential*, within the identity model presented in Chapter 4.
2. Test the overload caused by delegation, as it used for CCs presented in Chapter 5.
3. Test the performance associated to the CCs flow proposed in Chapter 5.
4. Test the feasibility of deploying an Hyperledger Aries Cloudagent controller in an Arduino device, so the SSI is validated on a really constrained IoT device.

6.1 Initial considerations

The proposed identity system for the IoTaaS (Chapter 4) can be implemented as long as it complies with W3C standards for DIDs and VCs ([23] and [24], respectively). However, some questions arise regarding the performance of the devices when the proposed protocol is implemented. Although some authors [42] have suggested that most *devices* are able to work with DIDs in terms of performance, we conducted some tests to determine whether it is also possible for *devices* to follow the proposed protocols and, specifically, to create and validate *consumer credentials*. IoT *devices* frequently depend on batteries, and it is important to check how SSI processes consume them. As providing an

accurate measure of battery usage is dependent on the target *device*, we analyzed CPU and RAM usage during the *consumer credential* creation and verification processes to evaluate the number of consumed resources, which provides more general conclusions.

As a reminder from Chapter 4, the actors participating in the identity model were: owner, consumer, certification entity and device, and their relationship are those represented in Figure 3-1. Each actor is implemented as a SSI agent and a SSI controller, so performance evaluation is done on these agents and controllers. In particular, the focus is put into the consumer credential exchange and the validation processes, which are the processes that deviate from the standard SSI flow. Regarding the *marketplace*, its implementation is expected in non-resource-constrained nodes that can be scaled with more resources or even with more nodes. Thus, we consider that, from a performance point of view, there are no specific issues. In addition, there are several possibilities to implement this component, so it is not possible to measure its performance as it depends on the final implementation. The most basic implementation could be a database and a web service. Relational databases can be appropriate for implementing this component, and so can non-relational databases, such as MongoDB [89]. Regarding the webservice, Apache [90] is the most widely used web server in the world, but other alternatives are also possible. MongoDB benchmark results are promising even with enormous amounts of data, performing even better than MySQL [86]. Traffic is normally managed through load balancers, such as Nginx [91], which allow to distribute the requests between different servers, then scaling the solution. Nginx can also substitute Apache as a web server.

Furthermore, it is important to figure out the impact of adding collaborative credentials in our setup. The protocol presented in Chapter 4 is designed to work without including delegation or collaboration, but as we have discussed in Chapter 5, they are useful to

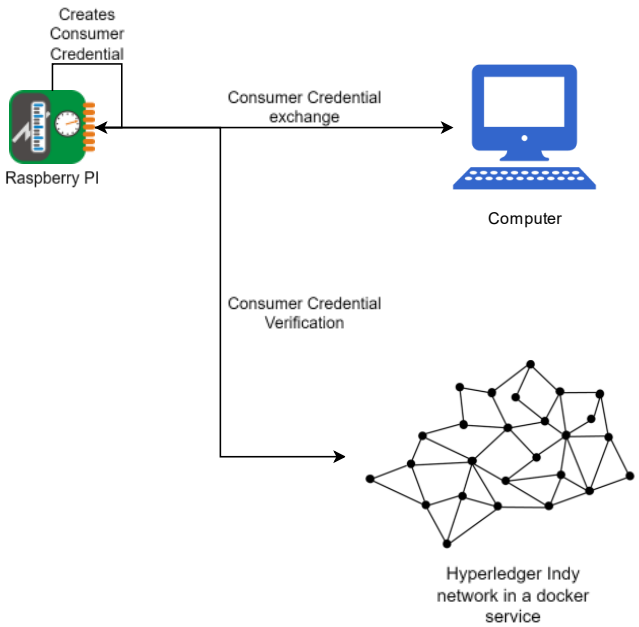

Figure 6-1: *Experimental setup for testbed 1*

expand the solution to fit more use cases. As it has been discussed, collaborative credentials make use of delegation. Consequently, it is interesting to know first if delegation adds an extra complication in terms of performance against the standard SSI flow. Furthermore, some performance metrics will be provided about CCs.

6.2 Testbed 1: Identity model

Testbed 1 aims to prove that the usage of *consumer credentials* does not impact substantially in the system in terms of performance. This implies testing the *consumer credential* creation, the *consumer credential* exchange and *the consumer credential verification*. To conduct the tests, a Raspberry Pi 2 B was used as IoT *device*. It is equipped with 1GB of RAM memory and uses a 900 MHz quad-core ARM Cortex-A7 CPU. A low-resource Lubuntu [92] operating system with a Hyperledger Indy client was installed in the device. Indy clients are currently under the Hyperledger Aries project, and the library used for testing has been the *Aries Framework JavaScript* project, available in Github [93]. JavaScript is a quite efficient language thanks to its asynchrony and allows the Raspberry Pi to achieve reasonable times because devices can perform other tasks while waiting for the next step in the consumer credential exchange and verification processes. Hence, JavaScript is appropriate for the implementation of Aries clients for IoTaaS, avoiding the blocking of the devices between these steps.

A wallet created in Raspberry Pi stores the *consumer credentials.* A wallet is a software is provided by Hyperledger Aries and uses several key libraries for its functionality. Its core functionality is managed by the indy-sdk, which is the foundation for managing VCs.

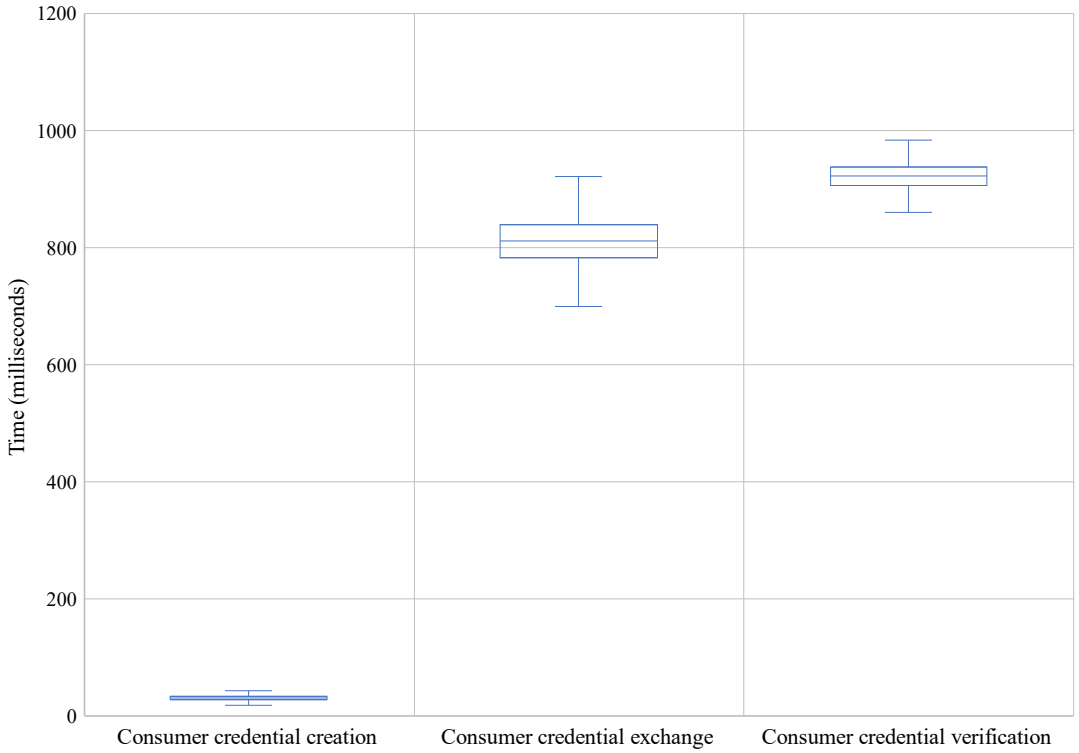

Figure 6-2: *Time (milliseconds) for each process evaluated in testbed 1*

Aries agents, provided by the *Aries Framework JavaScript* project are the software used to connect different SSI actors. Another computer, a Dell Latitude 5580 equipped with an Intel® Core™ i7-7600U and 8GM of RAM memory, located in the same network and connected by WiFi, plays the *consumer* role by first receiving a *consumer credential* and then sending a *VP* with this credential to the Raspberry Pi, which must validate it. Therefore, the hardware characteristics of this machine are irrelevant as all times are measured in the Raspberry Pi. Finally, a private Hyperledger Indy is used as a network with four nodes in four docker containers for implementing the VDR, deployed in the second machine.

Table 6-1: *Experimental results for the identity model*

	Measure	**Value**
Consumer credential creation	Time (avg)	31.07 ms
	Time (std)	4.80 ms
Consumer credential exchange	Time (avg)	811.95 ms
	Time (std)	73.69 ms
	CPU usage (avg)	18%
	RAM usage (avg)	150 MB
Consumer credential verification	Time (avg)	923.92 ms
	Time (std)	44.47 ms
	CPU usage (avg)	19%
	RAM usage (avg)	150 MB

In this setup, represented in Figure 6-1, we first measure the time spent by the *consumer credential* creation, the *consumer credential* exchange and the *consumer credential* verification, both within the *service consumption subprocess* (Section 4.3.3). Aries implements a several-step mechanism between *consumer* and *device*, similar to a TCP handshake, to complete these exchange and verification processes. We take 5,000 samples of every process and take the mean and standard deviation. In addition, we created 10,000 *consumer credentials* in the *device* to evaluate the time spent on this process. Tests were conducted using JEST [94] scripts, each of which took approximately 50 seconds to complete. Besides the time evaluation, RAM and CPU usage are measured in the *device* for *consumer credential* exchange and verification processes as well, but not for the consumer credential creation process, because its time is depreciable against the two others. CPU and RAM percentages were calculated using the average function among the four available logic cores in the Raspberry Pi and were measured during these 50 s. To establish a baseline, launching a JEST script that only loads the Aries libraries takes approximately 12 s and consumes 6MB of RAM memory and 4% of the CPU in average. The results are

shown in Table 6-1. Furthermore, Figure 6-2 shows the distribution for the average estimator of each subprocess.

The following conclusions can be drawn from these results. First, higher times are obtained in the processes for exchanging and verifying *consumer credentials*, rather than in creating them, being this process depreciable in terms of time spent. Second, regarding RAM and CPU usage, *consumer credential* exchange and verification are not resource-intensive tasks, so using devices with at least the capacity of the Raspberry Pi used in this experiment guarantees that the system will work without any problem. Consequently, we can conclude that the studied processes do not affect performance. However, according to the experiment, the minimum required RAM was approximately 150MB. Finally, we can conclude that the *consumer credential* exchange process is slightly faster than verification. The reason for this might be that the *device* needs to verify the validity of the *VP*, which adds extra complexity.

Network latency is another aspect to be considered when it comes to the final implementation, but it is always present and is not an SSI-related issue. However, it is true that the performed tests directly depend on network latency, so a *ping* command has been used to determine the base latency of the test network. To establish a baseline, each ICMP package in the deployed network takes an average of 0.28ms Round Time Trip (RTT), which is insignificant compared to the obtained times. Finally, regarding *consumers*, they can run SSI clients in a variety of hardware (laptops, smartphones, servers, etc), thus not implying performance issues because they are less constrained than a Raspberry Pi. Hence, the results have not been included in the present experiment because they fully depend on the chosen hardware.

It is important to note that these numbers can vary depending on factors such as network latency, intermediate network hops, and *device* capabilities. However, the results presented here are promising and show that most *devices* will be able to run the scheme without any issues.

6.3 Testbed 2: Delegation

This testbed aims to validate if the usage of delegation impacts on the performance of the SSI flow. To do so, a proof of concept using delegation has been created and it has been divided in seven steps, as shown below, and this has been compared against the standard SSI flow, that can be found in the Indy documentation. Thus, some results from the performance evaluation of an implementation of delegated credentials based on Hyperledger Indy with the Aries RFC approach are presented. Indy has been chosen instead of Aries so that the described problem of sequence numbers in Aries (see Section 5.2) is circumvented. This could be solved by using the modified Aries source code instead of

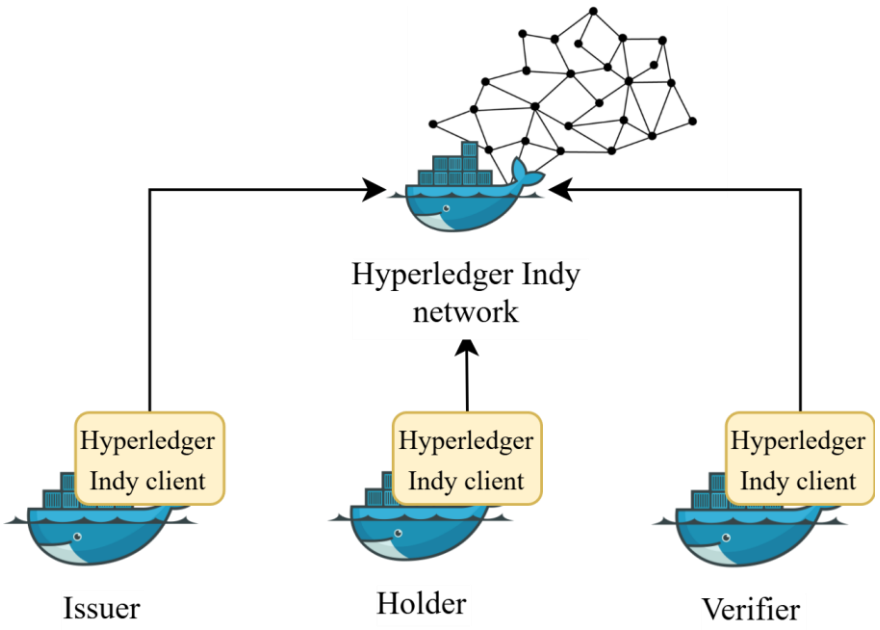

Figure 6-3: *Experimental setup for testbed 2*

using Indy, but then we could not extract accurate conclusions about the differences in time between the normal SSI flow, and the flow with delegation, which is the primary objective of this testbed, because if the code is not the same, it could affect the results.

Consequently, an Indy client in Python implementing the delegation has been developed and it has been deployed in three docker containers: issuer, holder, verifier. A private testing Indy network as VDR has also been deployed, containing four nodes running in a single docker container. The host VM with all docker containers is a Dell Latitude 5580 equipped with an Intel® Core™ i7-7600U and 8GM of RAM memory. This machine has been used instead of an IoT device because it is only required to measure the performance difference, and that means it does not matter what type of hardware is used. The software used could be run also on an IoT device. For simplicity, all actors are running in the same machine, so network latency will be very low because all systems are executed in the same machine. Figure 6-3 shows the setup for testbed 2.

The implementation includes these steps:

1) *A certification entity* creates and stores a Credential Schema and an associated Credential Definition;
2) The *certification entity* issues a delegated credential to a *manufacturer* with the Credential Definition, following the Aries RFC approach;

3) A *verifier* verifies the delegated credential from the *manufacturer;*
4) The *manufacturer* creates and stores a Credential Schema and a Credential Definition for a new VC;
5) The *manufacturer* creates a VP based on his delegated credential and sends the new VC to the *device*, which embeds the VP in an attribute;
6) The *verifier* verifies this new VC;
7) The *verifier* verifies the VP embedded in the VC.

Regarding the performance evaluation, the main interest was to analyse if the proposed delegation mechanisms impact in the performance when compared with a scenario without delegation. The latter scenario does not include an embedded VP into the VC and changing the step 7 by a standard verification against the *manufacturer*. The experiment has been conducted 400 times and the time spent to complete all the process with and without delegation is measured. The results are shown in Table 6-2. The Proof of Concept (PoC) with delegation corresponds to the seven steps described above, while the PoC without delegation corresponds to the normal Indy flow, which can be found in the Indy documentation. As it can be seen in Table 6-2, there are not significant differences in terms of performance (difference of 0.24 seconds in average), so any system supporting the normal Indy flow should support delegation as well with nearly insignificant impact in the performance. Furthermore, Figure 6-4 shows the times associated to both proofs of concept: with and without the delegation.

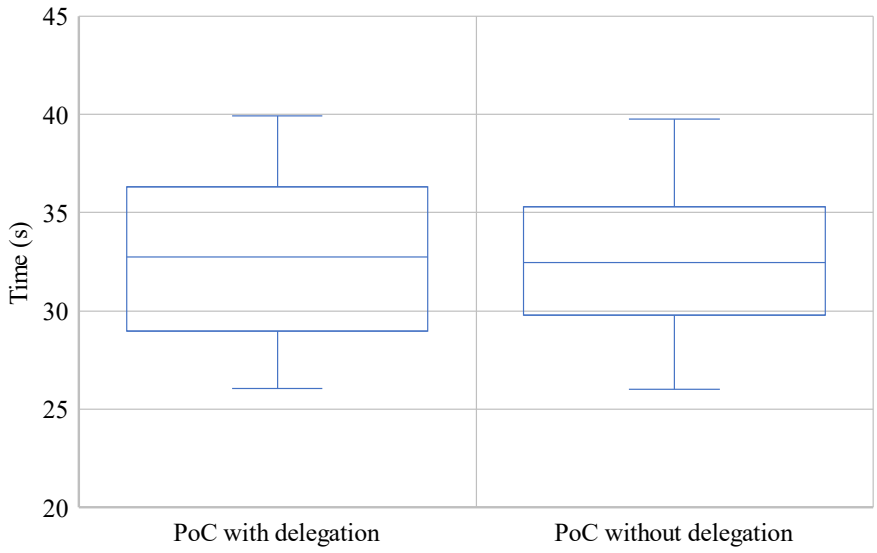

Figure 6-4: *Time (seconds) for the PoC with and without the delegation mechanism*

Table 6-2: *Experimental results for the comparison between scenarios with and without the usage of delegation*

	Measure	Value
PoC with delegation	Average	32.73 seconds
	Standard deviation	8.63 seconds
PoC without delegation	Average	32.49 seconds
	Standard deviation	7.53 seconds

6.4 Testbed 3: Collaborative Credentials

Now we are interested in assessing the feasibility in terms of performance for an implementation of the proposed model for CCs in Chapter 5. The implemented environment is as follows. A total of five modified Hyperledger Aries cloud agents have been installed in docker containers: one for the *initial coordinator*, one for the *verifier*, and three *group members*. Docker containers have been used to minimize the network latency in the experiment. Credential schemas and credential definitions are created by their issuers (*group members*). Five CLI controllers have been implemented using Python language, in five different docker containers. Furthermore, another controller has been installed in a Raspberry Pi 3. Therefore, we can measure how the protocol is affected when introducing real IoT devices. Figure 6-5 shows the implemented testbed. In addition, a decentralized Hyperledger Indy network has been installed out of the docker network. Therefore, we are simulating a completely decentralized testing environment.

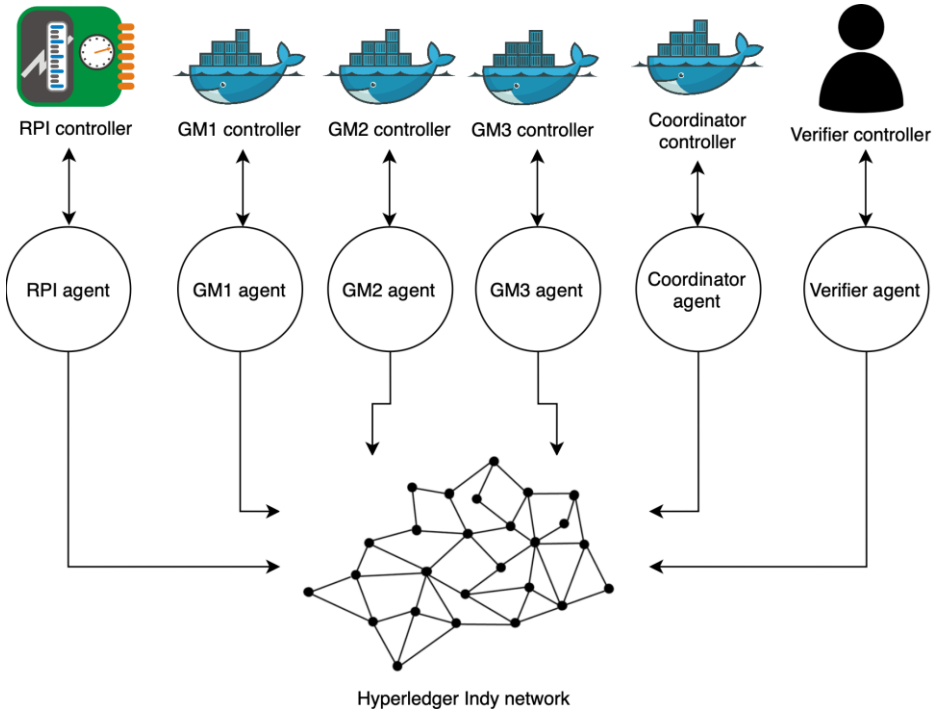

Figure 6-5: *Components of the implemented testbed for CCs*

We have measured the time it would take to complete the entire flow proposed in Chapter 5. We have launched the experiment 100 times and measured the mean and standard deviation values. Each time that we launch the experiment, the group is recreated, which means that a new *coordinator* must be chosen by consensus, and participants need to recreate the group. As explained, this is the worst-case scenario and provides us information about how well our protocol performs when there is a sudden change in the *coordinator* or any *group member*. Figure 6-6 shows the results split for every step of the protocol.

As shown in Figure 6-6, the discovery and onboarding processes clearly impact in performance as expected, as they involve sending a broadcast UDP package to the whole network, receive it by the *group members*, send it back to the *coordinator* and then the *coordinator* needs to onboard all *group members*. Therefore, the results confirm the intuitional idea that it is desirable to minimize the occurrence of the discovery and onboarding phases. As explained, a full mesh topology offers advantages against a tree topology because the discovery and onboarding must be done just once. Most of the time is spent during the onboarding, in particular, due to the method *accept_request* run in each *group member*. This method is provided by Hyperledger Aries and allows a *group member* to accept an invitation request from the *coordinator*. Therefore, it is related to Hyperledger Aries implementation instead of our protocol. Consequently, and based on the results, the protocol does not imply an additional complication against a normal Hyperledger Aries flow.

However, there is a significant difference between the Raspberry Pi and docker containers scenario (approximately 3 seconds) for the same number of *group members*, which clearly shows that the protocol is more affected by the computational power of the *group members* and network latency, rather than the steps of protocol.

Thus, we have also estimated the energy cost for the *group member* controller based on the tables of energy consumption for Raspberry, resulting in 480mA for the processing of requests generated by the command: *ab -n 100 -c* 10 (100 HTTP requests with a concurrency of 10 users), which is a considerable heavier task than running a *group member* controller. A Raspberry 3 consumes 260mA in idle mode (just switched on doing nothing). Therefore, the overload is maximum 220mA. “We have done this because mobility and unattendance are other features that characterize the IoT, meaning that these devices may be powered by batteries from time to time.

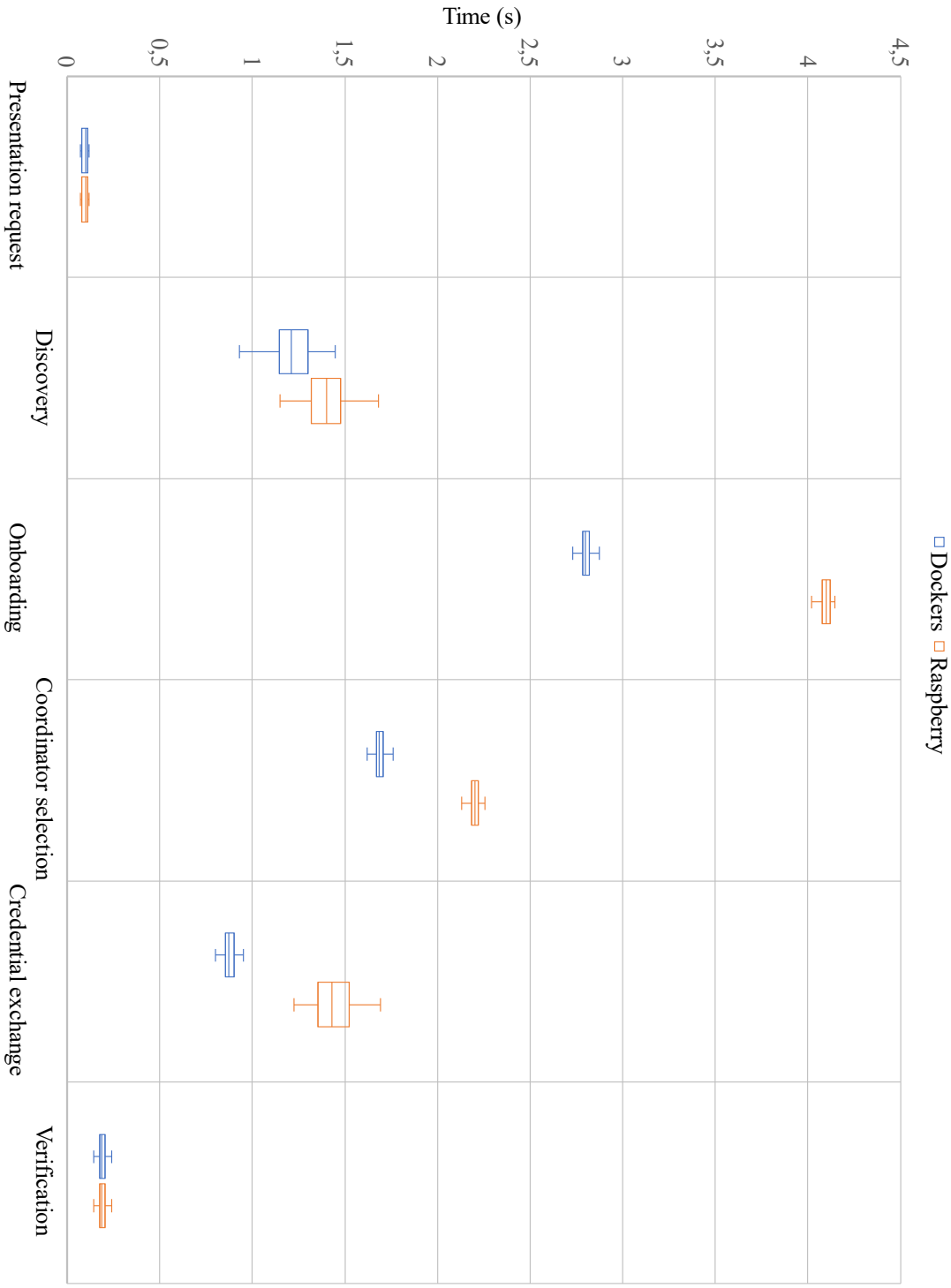

Figure 6-6: *Experimental results with 4 docker containers and with 3 docker containers and a Raspberry Pi*

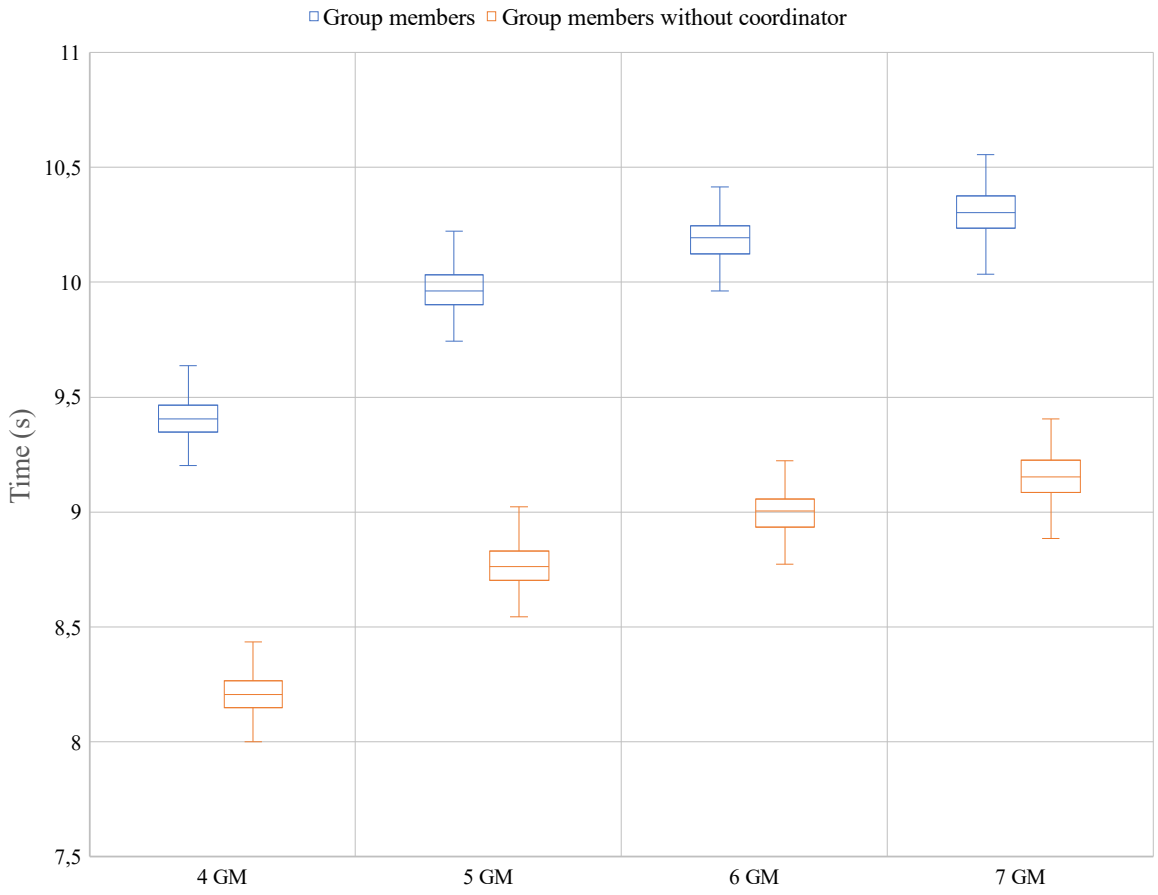

Figure 6-7: *Time (seconds) for different group members vs different group members in the scenario of polling with no coordinator*

We have measured that a whole cycle for a *group member* takes 1.2 seconds in average and adds a maximum consumption of 220mA. Thus, the consumption would be 73.3 uAh for every cycle. Each cycle is repeated each 60 seconds, as specified in the duration within the *CS*. On a normal battery of 5200 mAh, that would imply that this battery would last around 11.36 days with no charging, only considering the *group member* controller, which is acceptable in an environment such as the one considered. However, an optimal solution is to use a system that is connected to power, but where the connection is not reliable it can be backed up by simple or rechargeable batteries. This is mandatory if the hardware device is a Raspberry 3, which battery lasts a few hours just plugged in.

6.4.1 Comparison with polling strategy

We want to compare the performance of the CCs protocol against polling one to one of the *group members*. Removing the *coordinator* implies serious drawbacks. First, the *verifier* needs to discover the devices on his own, verify requirements individually and manage the changes in the structure of the group that may occur, then increasing his workload and energy cost considerably. If the *verifier* is a constrained device, it would impact the whole protocol. Secondly, the *coordinator* acts as a privacy proxy for the group and manages resource allocation to avoid race conditions. Third, an already-created group can be reused by other *verifiers*. Therefore, although the *coordinator* is an essential part in the proposed framework for CCs due to its advantages, we have repeated the experiment without it following the polling approach presented previously: the verifier periodically gets the identities of the members of the group, then poll every member of the group for credentials of its capabilities, and finally combine the obtained credentials to check if the requirements are met. The following changes have been made for this experiment: 1) the coordinator selection phase has been removed, 2) the verifier does the discovery and onboarding directly against the *group members*, 3) the verifier asks for a VP individually to the *group members* that have onboarded the group 4) *group members* send a VP to the verifier. We have measured the times for four to seven *group members* with the flow proposed is this work and the polling scenario without the *coordinator* (Figure 6-7). As expected, times are slightly better with polling (around 14% faster). By contrast and in addition to the aforementioned drawbacks, the number of messages managed by the verifier has substantially increased. With the normal flow, the verifier only sends 2 messages maximum, not depending on the number of *group members*. However, the polling alternative requires adding more messages to the verifier. Being *n* the number of *group members*, the verifier will need to send *1+3n* messages. If the number of *group members* is high, this is problematic as the verifier may be a constrained device. Therefore, adding the *coordinator* does not increase dramatically the time of the protocol, while it adds many advantages to it, being therefore justified to include it.

6.4.2 Scalability evaluation

We are also interested in discovering how the protocol scales when the number of *group members* increases, because ubiquity and myriad are also features of the IoT [2]. As the chosen topology is a tree topology with one level, the protocol has a *O(n)* complexity, being *n* the number of *group members*.

A scalability experiment has been carried out by using docker containers in one machine equipped with 48 Intel Xeon Gold 6146 CPUs running at 3.20GHz and 128 DDR4 RAM memory. The experiment has run with 10, 20 and 50 group members to check how the time and the number of messages increases with the number of group members.

Table 6-3 shows the total time in average and the standard deviation:

Table 6-3: *Total times for a different number of group members*

Total time	**Measure**	**Value**
10 group members	Average	12,49 seconds
	Standard deviation	1,24 seconds
20 group members	Average	22,33 seconds
	Standard deviation	2,26 seconds
50 group members	Average	50,41 seconds
	Standard deviation	8,98 seconds

Furthermore, times are disclosed for each method of the protocol and normalized by the number of group members in Figure 6-8.

The conducted scalability test shows that the time does not increase exponentially when the number of *group members* increases linearly, but also scales better than linearly when the number of *group members* increases. The conducted experiment has also shown that the discovery and onboarding phases are the most affected with an increase of the number of *group members*. This revalidates the argument that is desirable to minimize the occurrence of the discovery and onboarding phases.

6.5 Testbed 4: Evaluation with more constrained devices

Here we evaluate the installation of an Hyperledger Aries controller in an Arduino device, for the sake of testing the feasibility of using the proposed models when more constrained devices than Raspberry Pi are used and detecting possible workarounds to solve potential problems. The controller needs a private key to control its agent, which guarantees that there has been no impersonation of the controller.

Implementing an ACA-Py controller in an Arduino device has a drawback. ACA-Py uses webhooks to connect the agent (installed in the cloud) with the controller (installed in the end device, in this case, the Arduino).

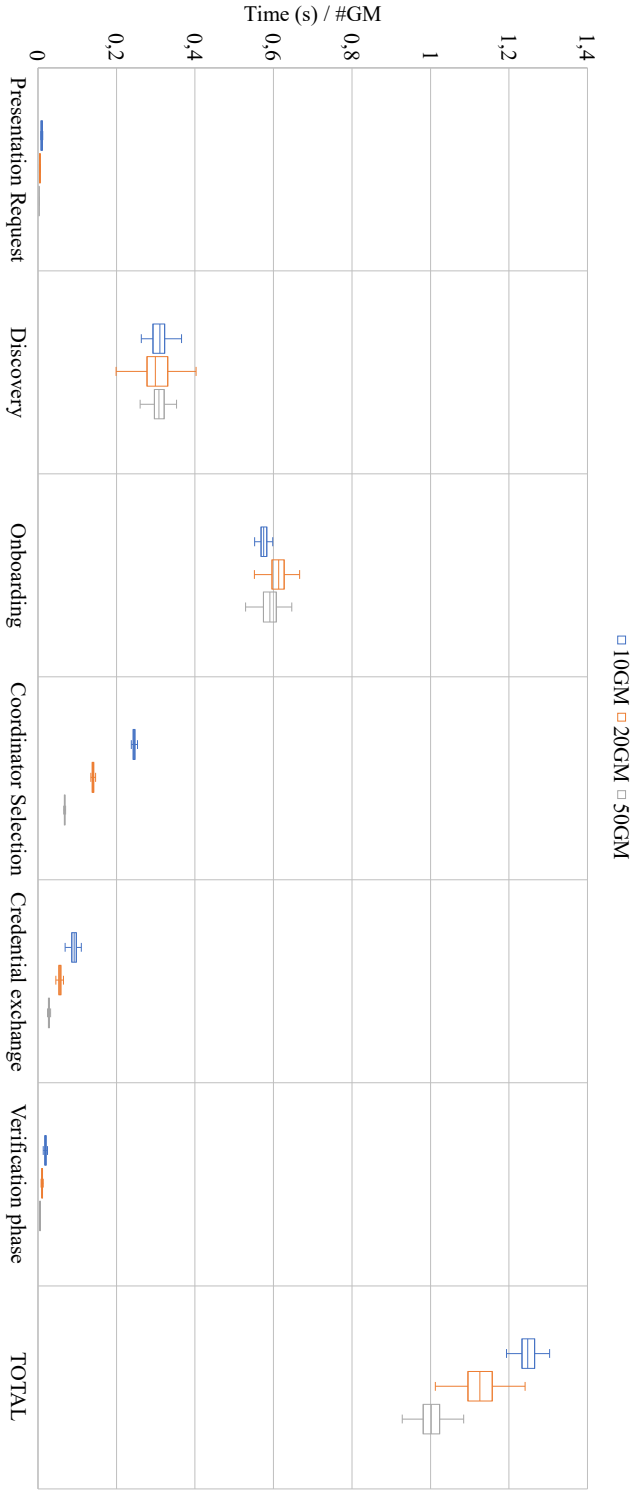

Figure 6-8: *Time (seconds) normalized for 10,20 and 50 group members in docker containers in a 48 Intel Xeon Gold 6146 CPUs*

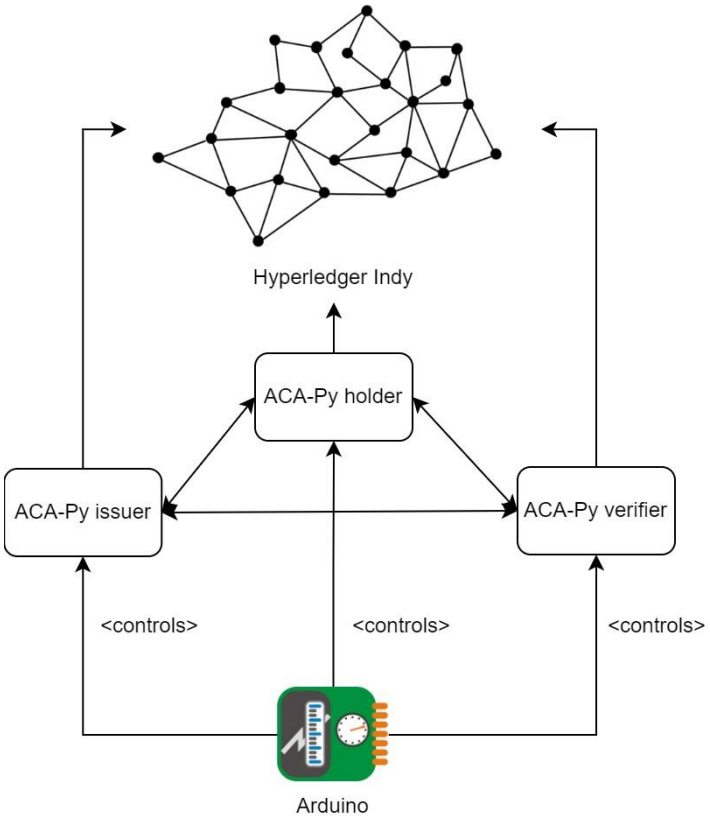

Figure 6-9: *Implemented testbed 4.*

Therefore, end devices need to deploy a webserver that listens from incoming communication to receive the information from the ACA-Py agent and take decisions according to the control logic. Regretfully, Arduino is not capable of deploying any webserv-er, what limits the implementation capability for ACA-Py. To address this challenge, we suggest two alternatives:

- Automatic mode. ACA-Py agents might be configured in an automatic mode that implies that there is no need to ask the controller what to do. In this case, the Arduino agent automatically accepts a VC and sends a VP when receives a PR, releasing the Arduino controller of any control action. However, this approach is very limited, as the end device cannot take any decision.
- Polling. In this case, the controller would periodically poll the ACA-Py Agent API to check if any action needs to be performed. This would allow the Arduino to decide whether accepting or not a VC and sending or not a VP as a response for a PR. This is the case that is proposed in this research work.

The decision between one alternative or the other will be use-case dependent and therefore there is not a better alternative over the other. However, a rule of thumb applies, which is: letting the ACA-Py agent manage most of the tasks, thus reducing to the minimum the amount of code that is run in the controller. This is congruent with IoT architectures where all the computation is taken out of the final devices as much as possible.

The device used for the performance evaluation is an Arduino SP32-C3-32S. It has a single core 32-bit RISC-V Microprocessor with clock frequency up to 160 MHz and 400 KB of SRAM. Most of the computation is done by the ACA-Py agent (cloud). The implemented testbed is as follows. Three Hyperledger Aries agents (ACA-Py) have been installed in three docker containers: issuer, holder and verifier: in the Dell computer presented above. To implement a VDR, an Hyperledger Indy network has been installed in the same machine, with four nodes in respective docker containers. Finally, an ACA-Py controller has been installed in an Arduino SP32-C3-32S. The Arduino controls the three ACA-Py agents (issuer, holder, verifier). As we are only interested in measuring times in the Arduino, it is sufficient to use just one that simulates the three controllers in the design. Figure 6-9 shows the implemented testbed.

The standard SSI methods are evaluated, which are Issue VC, Send PR, Generate VP, Verify VP; and also, polling. The implemented code, containing the three controllers, uses 877190 bytes (66%) of the program storage space. A timer is set in the Arduino and

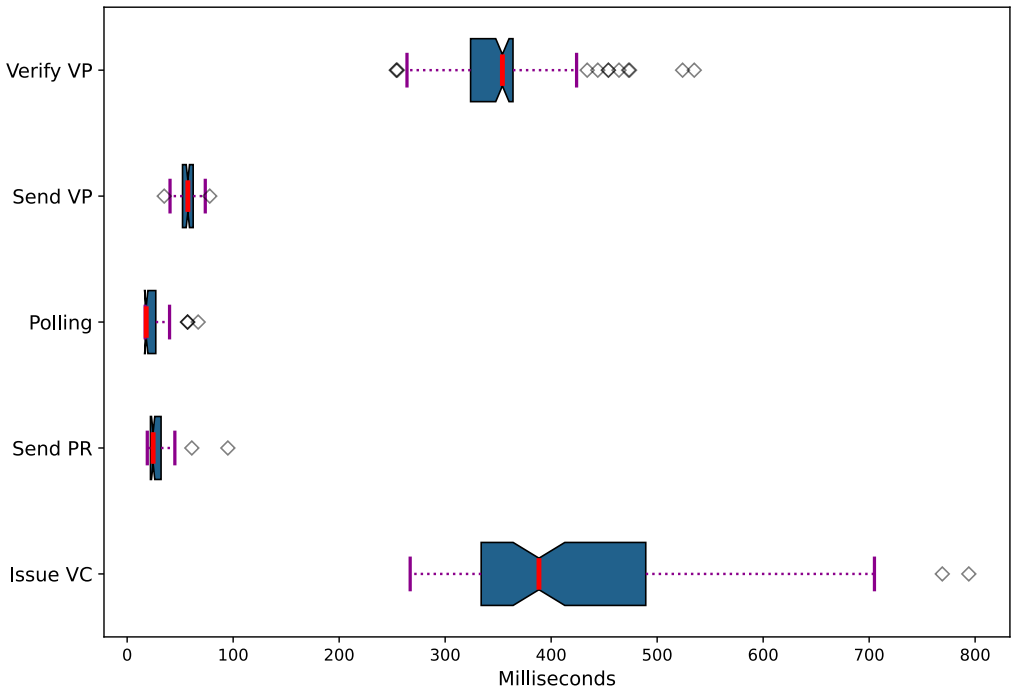

Figure 6-10: *Times for the different SSI methods.*

measures the time from the moment the method is called to the moment a response is received, which includes the computation done by the agent. For the experiment, the Arduino does the following steps. Each method has been launched 100 times and times are provided for each repetition 1) **Issue VC**: In step 1, the Arduino invokes the ACA-Py issuer agent to issue a VC to the holder agent. 2) **Send PR**: Arduino invokes the ACA-Py verifier agent to send a PR to the holder agent, asking for the VC issued in the step 1. 3) **Polling**: Arduino polls the ACA-Py holder agent for an incoming PR. 4) **Send VP**: Arduino invokes the ACA-Py holder agent to generate a VP that match the PR received in step 3 and sends it to the verifier agent. 5) **Verify VP**: Arduino invokes the ACA-Py verifier agent and verifies the received VP from step 4. Figure 6-10 shows the times for the studied SSI methods.

The results show that an ACA-Py controller can be implemented without any issues. Polling and sending a VP associated times only make sense when the controller is polling its corresponding Aries agent, because otherwise they are done automatically in the agent without the controller´s intervention. However, as shown, they are not time-consuming tasks. The tested SSI methods are done in less than 1 second (1000ms), mainly because the heavy computation is done in the ACA-Py agent instead of the controller. Issuing a VC is the most-consuming method, probably because it implies signing a VC and send it through the network. It also presents a big standard deviation, which probably depends on the network latency.

6.6 Summary of the chapter

The present chapter has demonstrated through several tests that the concepts and algorithms presented in the thesis can be implemented and do not imply a degradation in the performance.

To sum up, first of all the results of testbed 1 indicate that exchanging and verifying *consumer credentials* through a VP takes longer than creating them. While CPU and RAM usage are generally low, allowing most devices to perform these operations, a minimum of 150MB RAM is required. Parallel credential exchanges do not significantly impact resource usage, and the exchange process is marginally faster than verification, likely due to the added complexity of VP validation. Secondly, adding delegation to VCs does not impact the performance, as demonstrated through the testbed 2; and the proposed protocol for CCs, tested through testbed 3, demonstrates comparable performance to a standard Hyperledger Aries flow, with reasonable energy consumption. As group size increases, the total time grows sub-linearly, with discovery and onboarding phases being most affected. The use of a coordinator, while not significantly impacting protocol duration, provides substantial benefits, justifying its inclusion. Finally, through testbed 4, it has been demonstrated that a highly constrained device (Arduino SP32-C3-32S) can successfully run a ACA-Py controller. While implementation is feasible, it requires polling or the use of the automatic mode due to the Arduino's inability to support HTTP webhooks.

This page was intentionally left blank.

Chapter 7.

Conclusions and future work

Based on the conclusions provided for each chapter, we can draw the following general conclusions from this thesis.

IoTaaS formalization and sustainability benefits.

The concept of IoTaaS, although has been already discussed in industry and academia, has not yet been formally defined nor formalized. This thesis addressed this gap by providing a comprehensive formalization of the IoTaaS business model, identifying roles, interactions, and operational workflows. The analysis highlighted the potential of IoTaaS to contribute significantly towards sustainability objectives. By promoting the reuse of existing IoT resources and facilitating on-demand access to IoT devices, IoTaaS minimizes the environmental impact associated with the production of new devices and maximizes the utilization of already deployed hardware. Consequently, it encourages practices such as reuse, repurposing, and recycling, aligning with global sustainability goals and reducing overall economic and environmental costs.

Technological challenges of IoTaaS and importance of identity management.

Implementing the IoTaaS model introduces several technological challenges, particularly related to security, communication standards, semantic definitions, payment mechanisms, optimization processes and collaboration between devices. Of these, within security, identity management emerged as a fundamental challenge significantly affecting the security of IoTaaS. Given the myriad of interconnected devices and the necessity of robust authentication and authorization procedures, accurately managing device identities is paramount. In particular, the thesis highlighted that traditional, centralized identity management solutions fall short in scalability and security. In this context, the SSI emerges

as a promising paradigm for addressing identity management challenges in IoT environments and it offers advantages over traditional methods, particularly in enhancing security, privacy and data integrity within IoT ecosystems. The concept of CCs is introduced as a novel solution for managing group identities in IoT scenarios. CCs address scenarios requiring collaboration among multiple entities, which cannot be fully addressed by standard SSI implementations. The formal model for CCs, including their structure and lifecycle, provides a foundation for implementing group-based identity management in IoT.

Application of SSI to IoT and IoTaaS: feasibility and limitations.

The research successfully demonstrated that SSI schemes can feasibly address the identity management problem within IoT environments. Utilizing DIDs and VCs, SSI solutions significantly improve privacy, scalability, and device interoperability in IoTaaS scenarios. Specifically, SSI adoption in IoTaaS mitigates common IoT challenges, such as interdependence, diversity or unattendance, amongst other. However, despite these advantages, the thesis also identified specific limitations when applying SSI to IoT devices, particularly in resource-constrained scenarios. Limitations included computational overhead for cryptographic operations required by SSI (e.g., generating zero-knowledge proofs) or the need for proxy-based or hybrid approaches in extremely constrained devices. These limitations point to the necessity of further optimization or adaptation of SSI protocols to ensure broad applicability in diverse IoT contexts.

Delegation mechanisms and necessary adaptation in SSI.

As a preamble for the problem of CC, the thesis thoroughly investigated delegation mechanisms within SSI, which are essential for scenarios where identity credentials must be forwarded or delegated among entities. Current technologies reveal significant shortcomings, particularly related to implementation complexity with credential issuance and verification and interoperability among different SSI frameworks. A detailed analysis identified two primary delegation proposals: the W3C standard and the Hyperledger Aries RFC. The W3C delegation specification lacks support in some existing SSI frameworks and presented implementation complexity, because it requires that the wallet of the holder stores VCs which subject is not himself. This limitation is not present in the Aries RFC proposal, which stores embedded VPs instead of requiring holding VCs. Thus, the Aries RFC mechanism has greater practical compatibility with existing SSI infrastructure, as the concept of credential chaining through embedded VPs is well-supported in Aries-based frameworks, although it presents its own drawbacks, as some technologies relay on sequence numbers for the verification, which implies that some adaptations must be done to circumvent that. This has put into manifest that current technologies not always are fully compliant with the W3C standard for VCs and there is still a considerable gap between the

standard and its implementations. These findings also underline a clear necessity for adaptation and standardization of delegation mechanisms to fully realize the SSI potential.

Introduction and applicability of Collaborative Credentials (CCs).

CCs represent a novel extension to traditional SSI schemes by enabling credential usage through real-time collaboration among multiple entities. Standard SSI frameworks, as currently defined by the W3C standards for DIDs and VCs, typically focus on individual identity management. Within these traditional schemes, each VC is directly linked to one individual entity (holder) who manages and selectively discloses identity attributes independently. However, this individual-centric approach poses limitations when scenarios require real-time collaboration among multiple entities possessing distinct credentials.

CCs effectively address use cases where traditional SSI solutions, designed primarily for individual identity management, are insufficient. For instance, scenarios in healthcare, supply chain, and Industry 4.0 contexts where collaborative identification is required can greatly benefit from CCs. They also has applications in use cases for the IoTaaS. IoT ecosystems inherently involve interaction and collaboration between numerous devices, sensors, and other entities. Many IoT scenarios require coordinated authentication or authorization to carry out complex operations, such as real-time sensor measurements, aggregated data collection, or smart building management, tasks that single-device identities alone cannot efficiently manage. The CC framework enables groups of IoT devices or service providers to dynamically aggregate their credentials, providing collective authentication that simplifies operational workflows and enhances robustness against security breaches. Moreover, in the context of IoTaaS, the capability of CCs to dynamically form collaborative verification processes means that device groups can collaboratively fulfill verification requirements dynamically, increasing the flexibility, reliability, and trustworthiness of provided IoT services.

Formally speaking, CCs enable a group of individual credential holders to dynamically collaborate and collectively fulfill a verifier's requirements, thereby addressing complex scenarios involving multiple distinct identities. This expands SSI applicability into previously unsupported scenarios, providing substantial privacy benefits and enhancing overall flexibility and security in group identity management scenarios. Specifically, CCs are composed of a formal lifecycle, whose technical implementation, as provided in the thesis, includes a specific supportive framework built upon existing SSI technologies, demonstrating clearly how CCs can be realistically implemented and integrated into existing SSI infrastructure.

Feasibility and performance of CCs in IoT devices.

Finally, Chapter 6 provided empirical validation confirming that SSI and CCs implementations are feasible and practical within real IoT devices. The performance evaluation conducted demonstrated that IoT devices, even those with limited computational resources, could implement SSI schemes effectively without compromising performance. Experimental results indicated that delegation mechanisms and CC management processes could be conducted within acceptable timeframes, highlighting their suitability for deployment in realistic IoT scenarios. Tests revealed that CC-related operations could be appropriately optimized and executed efficiently. Hence, this thesis conclusively validates the practical feasibility and supports the integration of SSI and CC concepts within IoTaaS and broader IoT ecosystems.

Future work

The following lines of research could serve as future work in the field of IoTaaS and CCs: a) scalability optimization: deeper analysis of lightweight consensus algorithms and sidechains to support higher transaction rates; b) economic and pricing models: incorporation of supply-demand functions in the marketplace, incentives for quality and cooperation among device owners; c) enhanced privacy: application of advanced cryptography techniques (zk-SNARKs, homomorphic encryption, etc.) to ensure higher levels of confidentiality; d) large group support: expansion of CCs for scenarios involving thousands of devices and study of coordination protocols without a central coordinator; e) integration with, or inclusion at, international standards: cooperation with consortia such as W3C or ESSIF to unify specifications for the use of SSI and CCs in IoT devices.

Acronyms

ARM: Advanced Risk Machine

ASCII: American Standard Code for Information Interchange

AWS: Amazon Web Services

BTC: Bitcoin

CA: Certificates Authority

CAN: Controller Area Network

CC: Collaborative Credential

CLI: Command Line Interface

CoAP: Constraint Application Protocol

CPU: Central Processing Unit

CP-ABE: Cipher text Policy Attribute Based Encryption

dApp: Decentralised Application

DB: Database

DDR4: Double Data Rate Fourth Generation Synchronous Dynamic Random-Access Memory

DID: Decentralised Identifier

DLT: Distributed Ledger Technology

DLTS: Datagram Transport Layer Security

DoS: Denial of Service

ECU: Electronic Control Unit

ECC: Elliptic Curve Cryptography

GDPR: General Data Protection Regulation

GPS: Global Positioning System

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Secure

IaaS: Infrastructure-as-a-Service

ICMP: Internet Control Message Protocol

IIoT: Industrial Internet of Things

IoT: Internet of Things

IoTaaS: IoT-as-a-Service

IoV: Internet of Vehicles

IPFS: InterPlanetary File System

IT: Information Technology

JWS: JSON Web Signature

JWT: JSON Web Token

KP-ABE: Key text Policy Attribute Based Encryption

MQTT: Message Queuing Telemetry Transport

MQTT-S: MQTT for Sensor Networks

PaaS: Platform-as-a-Service

PGP: Pretty Good Privacy

PoC: Proof of Concept

PoS: Proof of Stake

PoW: Proof of Work

PR: Presentation Request

PSDR: Public Safety and Disaster Response

RC: Relationship Credential

RFC: Request For Feature

RAM: Random Access Memory

RTT: Round Time Trip

SaaS: Software-as-a-service

SLA: Service-Level Agreement

SMQTT: Secure Message Queue Telemetry Transport

SPoF: Single Point of Failure

SQL: Structured Query Language

SSO: Single Sign-On

SSI: Self-Sovereign Identity

TCP: Transmission Control Protocol

TEE: Trusted Execution Environment

TPM: Trusted Platform Module

UDP: User Datagram Protocol

URL: Uniform Resource Locator

UUID: Universally Unique Identifier

VC: Verifiable Credential

VDR: Verifiable Data Registry

VM: Virtual Machine

VP: Verifiable Presentation

ZKP: Zero-Knowledge Proof

This page was intentionally left blank.

References

[1] IBM. (2023, May 12). What is the Internet of Things (IoT)?. IBM. Available: https://www.ibm.com/think/topics/internet-of-things. Accessed: 03/01/2025.

[2] Zhou, W., Jia, Y., Peng, A., Zhang, Y., & Liu, P. (2018). The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. IEEE Internet of Things Journal, vol. 6(2), (pp. 1606-1616).

[3] Al-Qaseemi, S. A., Almulhim, H. A., Almulhim, M. F., & Chaudhry, S. R. (2016). IoT architecture challenges and issues: Lack of standardization. In 2016 Future technologies conference (FTC) (pp. 731-738). IEEE.

[4] ABIResearch. Key IoT Trends and Statistics Foreshadowing the Future. [Online]. Available: https://www.abiresearch.com/blogs/2024/01/30/current-and-future-iot-trends-and-statistics/#:~:text=Future%20IoT%20Statistics,at%20a%20CAGR%20of%2014%25. Accessed: 03/01/2025

[5] Greu, V. (2015). Facing IoT-The New Giant Wave of the Information and Communications Technologies Development. Romanian Distribution Committee Magazine, vol. 6(4), (pp. 18-25).

[6] Bag, S., Gupta, S., & Kumar, S. (2021). Industry 4.0 adoption and 10R advance manufacturing capabilities for sustainable development. International journal of production economics, vol. 231, (pp. 1-12).

[7] Khan, M. A., & Salah, K. (2018). IoT security: Review, blockchain solutions, and open challenges. Future Generation Computer Systems, vol. 82, (pp. 395-411).

[8] Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. IEEE Security & Privacy, vol. 9(3), (pp. 49-51).

[9] Kambourakis, G., Kolias, C., & Stavrou, A. (2017). The Mirai botnet and the IoT zombie armies. In MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM) (pp. 267-272). IEEE.

[10] Lu, Y., & Da Xu, L. (2018). Internet of Things (IoT) cybersecurity research: A review of current research topics. IEEE Internet of Things Journal, vol. 6(2), (pp. 2103-2115).

[11] Rajan, A., Jithish, J., & Sankaran, S. (2017). Sybil attack in IoT: Modelling and defenses. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 2323-2327). IEEE.

[12] Biometric Update.com. (2024). Identity verification scale and maturity to push average cost down. [Online]. Available: https://www.biometricupdate.com/202412/identity-verification-scale-and-maturity-to-push-average-cost-down. Accessed: 03/01/2025.

[13] Wieland, H., Hartmann, N. N., & Vargo, S. L. (2017). Business models as service strategy. Journal of the Academy of Marketing Science, 45, 925-943.

[14] Goyal, S. (2013). Software as a service, platform as a service, infrastructure as a service– a review. International Journal of Computer Science & Network Solutions, 1(3), 53-67.

[15] Au-Yong-Oliveira, M., Marinheiro, M., & Tavares, J. A. C. (2020). The Power of Digitalization: The Netflix Story. In World Conference on Information Systems and Technologies (pp. 590-599). Springer, Cham.

[16] Mathew, S., & Varia, J. (2014). Overview of amazon web services. Amazon Whitepapers, vol. 105(1), 22.

[17] Deng, D. J., Pang, A. C., & Hanzo, L. (2019). Recent Advances in IoT as a Service (IoTaas 2017). Mobile Networks and Applications, 24(3), (pp. 721-723).

[18] Brincat, A. A., Pacifici, F., & Mazzola, F. (2019). IoT as a service for smart cities and nations. IEEE Internet of Things Magazine, vol. 2(1), (pp. 28-31).

[19] DIF – Decentralized Identity Foundation. Principles and Characteristics of Self Sovereign Identity. [Online]. Available: https://decentralized-id.com/self-sovereign-identity/characteristics/. Accessed: 03/01/2025.

[20] Sharif, A., Ranzi, M., Carbone, R., Sciarretta, G., Marino, F. A., & Ranise, S. (2022). The eIDAS regulation: a survey of technological trends for European electronic identity schemes. Applied Sciences, 12(24), 12679.

[21] Council of the European Union. (2021). Council Recommendation (EU) 2021/946 of 14 June 2021 on the establishment of the European Child Guarantee. EUR-Lex. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32021H0946. Accessed: 03/01/2025.

[22] DIF – Decentralized Identity Foundation. European Blockchain Services Infrastructure (EBSI) and the eSSIF. [Online]. Available: https://decentralized-id.com/government/europe/eu/ebsi-essif/. Accessed: 03/01/2025.

[23] Reed, D., Sporny, M., Longley, D., Allen, C., Grant, R., Sabadello, M., & Holt, J. (2020). Decentralized identifiers (DIDs) v1. 0. Draft Community Group Report.

[24] World Wide Web Consortium. (2024). Verifiable Credentials Data Model 2.0: Expressing Verifiable Information on the Web. [Online]. Available: https://www.w3.org/TR/vc-data-model-2.0/. Accessed: 03/01/2025.

[25] Jones, M. (2015). JSON web token (JWT). Internet Engineering Task Force (IETF) RFC, 7519.

[26] W3C. (2021). Linked Data Security Working Group Charter. [Online]. Available: https://lists.w3.org/Archives/Public/public-credentials/2021May/att-0082/2021-Linked-Data-Security-WG-Charter.pdf. Accessed: 03/01/2025.

[27] Camenisch, J., & Lysyanskaya, A. (2001). An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20 (pp. 93-118). Springer Berlin Heidelberg.

[28] Kulabukhova, N. V. (2019). Zero-knowledge Proof in Self-Sovereign Identity. In Proceedings of the 27th International Symposium Nuclear Electronics and Computing (NEC'2019) (pp. 381-385).

[29] Nakamoto, S. (2019). Bitcoin: A peer-to-peer electronic cash system. Manubot.

[30] Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., & Qijun, C. (2017). A review on consensus algorithm of blockchain. In 2017 IEEE international conference on systems, man, and cybernetics (SMC) (pp. 2567-2572). IEEE.

[31] Venkatesan, K., & Rahayu, S. B. (2024). Blockchain security enhancement: an approach towards hybrid consensus algorithms and machine learning techniques. Scientific Reports, 14(1), 1149.

[32] Asir, T. R. G., Manohar, H. L., Anandaraj, W., & Sivaranjani, K. N. (2016). IoT as a Service. In 2016 International Conference on innovations in Information, Embedded and Communication Systems (ICIIECS) (pp. 1093-1096).

[33] Buzachis, A., Fazio, M., Galletta, A., Celesti, A., & Villari, M. (2019). Infrastructureless IoT-as-a-service for public safety and disaster response. In 2019 7th

International Conference on Future Internet of Things and Cloud (FiCloud) (pp. 133-140). IEEE.

[34] Świątek, P., & Rucinski, A. (2013). IoT as a service system for eHealth. In 2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013) (pp. 81-84). IEEE.

[35] Giacobbe, M., Di Pietro, R., Minnolo, A. L., & Puliafito, A. (2018). Evaluating information quality in delivering IoT-as-a-service. In 2018 IEEE international conference on Smart Computing (SMARTCOMP) (pp. 405-410). IEEE.

[36] Borsatti, D., Davoli, G., Cerroni, W., & Raffaelli, C. (2021). Enabling Industrial IoT as a Service with Multi-Access Edge Computing. IEEE Communications Magazine, vol. 59(8), (pp. 21-27).

[37] Pérez, J. L., & Carrera, D. (2015). Performance characterization of the Servioticy API: an IoT-as-a-Service data management platform. In 2015 IEEE First International Conference on Big Data Computing Service and Applications (pp. 62-71). IEEE.

[38] Fedrecheski, G., Rabaey, J. M., Costa, L. C., Ccori, P. C. C., Pereira, W. T., & Zuffo, M. K. (2020). Self-Sovereign Identity for IoT environments: a perspective. In 2020 Global Internet of Things Summit (GIoTS) (pp. 1-6). IEEE.

[39] S. Foundation. (2020). Self-Sovereign Identity and IoT. [Online]. Available: https://sovrin.org/wp-content/uploads/SSI-and-IoT-whitepaper.pdf. Accessed: 03/01/2025.

[40] Bartolomeu, P. C., Vieira, E., Hosseini, S. M., & Ferreira, J. (2019). Self-Sovereign Identity: Use-cases, technologies, and challenges for industrial IoT. In 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 1173-1180). IEEE.

[41] Mahalle, P. N., Shinde, G., & Shafi, P. M. (2020). Rethinking decentralised identifiers and verifiable credentials for the Internet of Things. In Internet of Things, Smart Computing and Technology: A Roadmap Ahead (pp. 361-374). Springer, Cham.

[42] Kortesniemi, Y., Lagutin, D., Elo, T., & Fotiou, N. (2019). Improving the privacy of IoT with Decentralised Identifiers (DIDs). Journal of Computer Networks and Communications, vol. 2019. (pp. 1-10).

[43] Berzin, O., Ansay, R., Kempf, J., Sheikh, I. & Hendel, D. (2021). The IoT Exchange. [Online]. Available: https://arxiv.org/abs/2103.12131. Accessed: 03/01/2025.

[44] Weingärtner, Tim. (2021). Identity of Things: Applying concepts from Self Sovereign Identity to IoT devices. The Journal of The British Blockchain Association, vol. 4. (pp. 1-7).

[45] Gebresilassie, S. K., Rafferty, J., Morrow, P., Chen, L. L., Abu-Tair, M., & Cui, Z. (2020). Distributed, Secure, Self-Sovereign Identity for IoT Devices. In 2020 IEEE 6th World Forum on Internet of Things (WF-IoT) (pp. 1-6). IEEE.

[46] Popov, S. (2018). The tangle. White paper, vol. 1(3), 30.

[47] Luecking, M., Fries, C., Lamberti, R., & Stork, W. (2020). Decentralized identity and trust management framework for Internet of Things. In 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1-9). IEEE.

[48] Chohan, U. W. (2021). The double spending problem and cryptocurrencies. Available at SSRN 3090174.

[49] Abishek, S. V., Priyan, G. D., More, S., & Suganthan, A. (2019). 'IoT based smart band for biometric authentication using blockchain technology. Int. J. Recent Technol. Eng, vol. 7(5c), (pp. 175-181).

[50] Theodouli, A., Moschou, K., Votis, K., Tzovaras, D., Lauinger, J., & Steinhorst, S. (2020). Towards a Blockchain-based Identity and Trust Management Framework for the IoV Ecosystem. In 2020 Global Internet of Things Summit (GIoTS) (pp. 1-6). IEEE.

[51] Windley, P., & Reed, D. (2018). Sovrin: A protocol and token for Self-Sovereign Identity and decentralized trust. Whitepaper, The Sovrin Foundation.

[52] Naik, N., & Jenkins, P. (2020). uPort open-source identity management system: An assessment of Self-Sovereign Identity and user-centric data platform built on blockchain. In 2020 IEEE International Symposium on Systems Engineering (ISSE) (pp. 1-7). IEEE.

[53] Veramo - Performant and modular APIs for Verifiable Data and SSI. [Online]. Available: https://veramo.io/. Accessed: 03/01/2025.

[54] Ali, M., Shea, R., Nelson, J., & Freedman, M. J. (2017). Blockstack Technical Whitepaper. Blockstack PBC, October, 12.

[55] Veres One - A Globally Interoperable Blockchain for Identity. [Online]. Available: https://veres.one/. Accessed: 03/01/2025.

[56] Jolocom - A Decentralized, Open Source Solution for Digital Identity and Access Management. [Online]. Available: https://www.alchemy.com/dapps/jolocom. Accessed: 03/01/2025.

[57] Hyperledger Aries. [Online]. Available: https://github.com/hyperledger/aries. Accessed: 03/01/2025.

[58] Hyperledger Aries Cloud Agent. [Online]. Available: https://github.com/hyperledger/aries-cloudagent-python. Accessed: 03/01/2025.

[59] Davie, M., Gisolfi, D., Hardman, D., Jordan, J., O'Donnell, D., & Reed, D. (2019). The trust over IP stack. IEEE Communications Standards Magazine, vol. 3(4), (pp. 46-51).

[60] Curren, S., Looker, T., & Terbu, O. (2022). Didcomm messaging. s Draft, 1. [Online]. Available: https://identity.foundation/didcomm-messaging/spec/. Accessed: 03/01/2025.

[61] Buterin, V. (2014). A next-generation smart contract and decentralized application platform. white paper, vol. 3(37).

[62] Benet, J. (2014). IPFS-content addressed, versioned, P2P file system. arXiv preprint arXiv:1407.3561.

[63] Fotiou, N., Siris, V. A., Xylomenos, G., & Polyzos, G. C. (2022). IoT Group Membership Management Using Decentralized Identifiers and Verifiable Credentials. Future Internet, vol. 14(6), 173.

[64] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., & Shacham, H. (2009). Randomizable proofs and delegatable anonymous credentials. In Annual International Cryptology Conference (pp. 108-125). Springer, Berlin, Heidelberg.

[65] Chase, M., Kohlweiss, M., Lysyanskaya, A., & Meiklejohn, S. (2013). Malleable signatures: Complex unary transformations and delegatable anonymous credentials. Cryptology ePrint Archive.

[66] Camenisch, J., Drijvers, M., & Dubovitskaya, M. (2017). Practical UC-secure delegatable credentials with attributes and their application to blockchain. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 683-699).

[67] Hardman D. & Harchandani, L. Aries RFC 0104: Chained Credentials. [Online]. Available: https://github.com/hyperledger/aries-rfcs/tree/main/concepts/0104-chained-credentials. Accessed: 03/01/2025.

[68] Zheng, W. (2006). The business models of e-marketplace. Communications of the IIMA, vol. 6(4), 1.

[69] Lu, R. (2018). A new communication-efficient privacy-preserving range query scheme in fog-enhanced IoT. IEEE Internet of Things Journal, vol. 6(2), (pp. 2497-2505).

[70] Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008). MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08) (pp. 791-798). IEEE.

[71] Singh, M., Rajan, M. A., Shivraj, V. L., & Balamuralidhar, P. (2015). Secure MQTT for Internet of Things (IoT). In 2015 fifth international conference on communication systems and network technologies (pp. 746-751). IEEE.

[72] Bamert, T., Decker, C., Elsen, L., Wattenhofer, R., & Welten, S. (2013). Have a snack, pay with Bitcoins. In IEEE P2P 2013 Proceedings (pp. 1-5). IEEE.

[73] Mihaylov, M., Jurado, S., & Avellana, N. (2014). Virtual currency for trading of renewable energy in smart grids. In Proceedings of the International Conference on the European Energy Market, EEM (pp. 1-6).

[74] Alam, M. T., Li, H., & Patidar, A. (2015). Bitcoin for smart trading in smart grid. In The 21st IEEE International Workshop on Local and Metropolitan Area Networks.

[75] Brody, P., Pureswaran, V., Panikkar, S., & Nair, S. (2015). Empowering the edge Practical insights on a decentralized Internet of Things. IBM Institute for Business Value. Technical report.

[76] Zhu, X., & Badr, Y. (2018). Identity management systems for the Internet of Things: a survey towards blockchain solutions. Sensors, vol. 18(12), 4215.

[77] Sabt, M., Achemlal, M., & Bouabdallah, A. (2015). Trusted execution environment: What it is, and what it is not. In 2015 IEEE Trustcom/BigDataSE/Ispa (Vol. 1, pp. 57-64). IEEE.

[78] Kinney, S. L. (2006). Trusted platform module basics: using TPM in embedded systems. Elsevier.

[79] Voigt, P., & Von dem Bussche, A. (2017). The EU General Data Protection Regulation (GDPR). A Practical Guide, 1st Ed., Cham: Springer International Publishing, 10, 3152676.

[80] Gupta, P., Kanhere, S., & Jurdak, R. (2019). A decentralized IoT data marketplace. arXiv preprint arXiv:1906.01799.

[81] Sermersheim, J. (2006). Lightweight directory access protocol (LDAP): The protocol. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4511.html. Accessed: 03/01/2025.

[82] Lim, S., Rhie, M. H., Hwang, D., & Kim, K. H. (2021). A subject-centric credential management method based on the verifiable credentials. In 2021 International Conference on Information Networking (ICOIN) (pp. 508-510). IEEE.

[83] Li Park, C. S., & Park, W. S. (2018). A group-oriented DTLS handshake for secure IoT applications. IEEE Transactions on Automation Science and Engineering, vol. 15(4), (pp. 1920-1929).

[84] Lynch, N. A. (1996). Distributed algorithms. Morgan Kaufmann Publishers.

[85] Al-Maaitah, S., Qatawneh, M., & Quzmar, A. (2021). E-Voting System Based on Blockchain Technology: A Survey. In 2021 International Conference on Information Technology (ICIT) (pp. 200-205).

[86] IEEE Jose, B., & Abraham, S. (2020). Performance analysis of NoSQL and relational databases with MongoDB and MySQL. Materials today: PROCEEDINGS, 24, 2036-2043.

[87] Fett, D., Küsters, R., & Schmitz, G. (2016). A comprehensive formal security analysis of OAuth 2.0. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 1204-1215).

[88] Fett, D., Küsters, R., & Schmitz, G. (2017). The web SSO standard OpenID connect: In-depth formal security analysis and security guidelines. In 2017 IEEE 30th Computer Security Foundations Symposium (CSF) (pp. 189-202). IEEE.

[89] Győrödi, C., Győrödi, R., Pecherle, G., & Olah, A. (2015). A comparative study: MongoDB vs. MySQL. In 2015 13th International Conference on Engineering of Modern Electric Systems (EMES) (pp. 1-6). IEEE.

[90] Laurie, B., & Laurie, P. (2003). Apache: The definitive guide (3rd ed.). O'Reilly Media, Inc. ISBN: 978-0596002039.

[91] Nedelcu, C. (2013). Nginx HTTP Server. Packt Publishing.

[92] Medina, F. A. B., Menjívar, T. G., & Flores, E. O. C. (2019). Lubuntu. SolTic UGB, vol. 3(2), (pp. 21-30).

[93] Aries Framework Javascript. [Online]. Available: https://github.com/hyperledger/aries-framework-javascript. Accessed: 03/01/2025.

[94] Delightful JavaScript Testing. [Online]. Available: https://jestjs.io/. Accessed: 03/01/2025.

This page was intentionally left blank.

ANEXO I:

Resumen de la Tesis Doctoral

Tecnologías habilitadoras para mejorar la seguridad de un modelo de negocio de IoT como Servicio (IoTaaS)

Resumen

El modelo de negocio IoT-as-a-Service (IoTaaS) ya ha sido identificado por algunas personas tanto del mundo empresarial como académico, pero no ha sido definido formalmente. IoTaaS propone ofrecer dispositivos IoT bajo demanda, implicando considerables ahorros de costes y optimización de recursos. Además, permite que diferentes aplicaciones reutilicen los dispositivos existentes. Sin embargo, este modelo de negocio está asociado con diferentes desafíos tecnológicos que deben abordarse, uno de los cuales es el problema de la identidad. La gestión de identidad es fundamental para lograr seguridad. Además, la Industria 4.0 está destinada a modernizar los procesos industriales tal como los conocemos hoy. Esta modernización va de la mano con la digitalización de la industria y la necesidad de identificar digitalmente los diferentes dispositivos involucrados en el proceso de fabricación. En consecuencia, como contribuciones, esta tesis pretende formalizar el IoTaaS, identificar sus desafíos tecnológicos desde una perspectiva de seguridad y proponer un modelo de identidad basado en SSI para el IoTaaS.

Centrándose en la gestión de identidad, los esquemas de identidad auto-soberana (SSI) han demostrado proporcionar mejor privacidad y escalabilidad que los paradigmas de identidad tradicionales, lo cual es especialmente importante en el IoT debido a sus características. Las credenciales verificables y los Identificadores Descentralizados, que forman parte del concepto de identidad auto-soberana, permiten la identificación y

caracterización descentralizada de los dispositivos (comúnmente dispositivos IIoT) que componen la Industria 4.0. Sin embargo, algunos casos de uso en la Industria 4.0 no pueden modelarse con esquemas SSI estándar. A pesar de que las credenciales delegadas se han definido en el estándar W3C para credenciales verificables, las tecnologías actuales presentan algunas limitaciones importantes que las hacen no implementables. Esta tesis analiza estas limitaciones en el contexto del problema de construir credenciales delegadas para la Industria 4.0, y propone una alternativa basada en un RFC de Hyperledger Aries, superando estas limitaciones.

Basado en el problema anterior de las credenciales delegadas, surge un nuevo problema. Los protocolos y procedimientos SSI estándar actuales solo consideran que los individuos almacenan su propia identidad, sin proporcionar una solución precisa para la gestión de identidad de grupos donde los participantes pueden usar credenciales de diferentes identidades y colaborar para cumplir con un conjunto de requisitos del verificador. La identificación de grupos se ha identificado como otro desafío para el IoTaaS. En consecuencia, la tesis también introduce el concepto de Credenciales Colaborativas (CCs) para formalizar los procedimientos de gestión de identidad que modelan la colaboración dentro de un grupo de participantes. Las CCs permiten aprovechar casos de uso que requieren colaboración que no pueden resolverse con credenciales verificables SSI estándar, aumentan la privacidad de los participantes del grupo y permiten el desarrollo de un marco de software que cualquier verificador/titular podría usar para generar una aplicación genérica.

En resumen, en esta tesis se analiza formalmente el modelo de negocio IoTaaS, identificando y detallando sus principales desafíos tecnológicos. Además, se aborda el problema de identidad de este modelo de negocio y se propone un sistema de gestión de identidad basado en SSI, que cumple con los estándares existentes de la W3C. Como parte del problema de identidad, también se analizan los esquemas de delegación y el uso de CCs. Finalmente, se evalúa el modelo de identidad del IoTaaS en términos de rendimiento, así como algunas pruebas de implementación con respecto al uso de delegación y CCs.

Introducción y motivación

La tesis comienza analizando el crecimiento exponencial del IoT y de la búsqueda de modelos de negocio innovadores. Concretamente, la idea de “IoT-as-a-Service” (IoTaaS) se perfila como una alternativa que permite ofrecer dispositivos IoT bajo demanda a distintos consumidores, optimizando tanto los recursos como los costes.

Aunque el IoTaaS promete beneficios claros, como la reutilización de dispositivos y la posibilidad de compartir datos entre múltiples aplicaciones, conlleva también desafíos tecnológicos significativos. Entre ellos, destaca la necesidad de un sistema robusto de gestión de la identidad y la seguridad, ya que cualquier uso indebido puede comprometer

los datos o servicios que brindan estos dispositivos. Este trabajo de investigación justifica, por tanto, la importancia de definir formalmente el modelo de negocio, de caracterizar sus principales actores y de proponer soluciones de identidad y seguridad que sean escalables y confiables.

Asimismo, el documento expone la relevancia de aplicar el paradigma de SSI a estos contextos de IoT, pues la descentralización y el control de la identidad por parte del sujeto de identidad (tanto si es un usuario humano como un dispositivo) resultan particularmente útiles al evitar la dependencia de terceros y grandes repositorios centralizados de datos. La tesis explica que la SSI no solo resulta ventajosa frente a sistemas de identidad tradicionales (centralizados, federados, etc.), sino que también responde a características esenciales de la IoT: multitud de dispositivos, heterogeneidad de fabricantes, recursos de cómputo limitados y altas necesidades de privacidad.

La investigación también detecta que la gestión de identidad para grupos de dispositivos o colectivos de usuarios es un aspecto poco atendido por los estándares actuales de SSI. Para llenar este vacío, en la tesis se introduce el concepto de "Collaborative Credentials" (CCs), o Credenciales Colaborativas, que extiende los principios de la SSI a escenarios colaborativos donde múltiples participantes deben coordinarse y demostrar colectivamente ciertos atributos.

El IoT ha experimentado un crecimiento exponencial en los últimos años, revolucionando diversos sectores como la salud, la fabricación y las ciudades inteligentes. Sin embargo, la integración del IoT en modelos de negocio, especialmente como servicio (IoTaaS) presenta desafíos y oportunidades únicas. Esta tesis aborda la necesidad crítica de un modelo de negocio IoTaaS seguro y escalable.

El paradigma IoTaaS ofrece un enfoque flexible para implementar soluciones IoT bajo demanda. No obstante, existe una falta de estudios que definan claramente el modelo, los roles y las responsabilidades de los diversos actores involucrados. Además, la gestión de identidades es un pilar fundamental de la seguridad en entornos IoT. No se puede lograr un nivel de seguridad adecuado sin una gestión de identidades apropiada.

En consecuencia, esta tesis tiene como objetivo desarrollar un modelo de identidad integral para el IoTaaS, abordando los requisitos y restricciones únicos de los dispositivos y servicios IoT. Tras analizar diferentes esquemas de gestión de identidades para IoT, se ha concluido que la SSI presenta claras ventajas frente a los sistemas tradicionales.

La SSI aspira a que las identidades digitales sean controladas por los propios titulares (*holder*), quienes almacenan sus credenciales (por ejemplo, en una "*wallet*"). Es clave el uso de DIDs (Identificadores Descentralizados) como identificadores únicos de cada entidad, que pueden ser verificados en un registro descentralizado (DLT o equivalente). Además, las Credenciales Verificables (VCs) son equivalentes a "credenciales digitales"

firmadas criptográficamente por emisores confiables. Cuando un usuario (o dispositivo) presenta sus credenciales para su verificación por un verificador (*verifier*), genera una Presentación Verificable (VP), que incluye pruebas criptográficas para demostrar la validez de dichas credenciales, respetando al máximo la privacidad (p.ej., usando *zero-knowledge proofs*).

El registro descentralizado donde se inscriben los DIDs y metadatos (VDR, o Registro de Datos Verificable) se puede implementar mediante blockchain u otros *Distributed Ledger Technology* (DLT), otorgando inmutabilidad y evitando un punto central de fallo y de control. Se discuten diversas alternativas (Bitcoin, Ethereum, Hyperledger Indy, IOTA) y se hace hincapié en la necesidad de mecanismos de consenso que permitan alta escalabilidad, bajas comisiones y seguridad frente a ataques.

Justificación del uso de SSI para IoT

La Identidad Auto-Soberana (SSI) ofrece numerosas ventajas sobre otras alternativas de identidad digital para IoT:

1. Mejora la privacidad al utilizar DIDs como identificadores para dispositivos IoT.
2. Elimina los "silos de identidad", reduciendo el riesgo de ataques.
3. Simplifica la gestión de identidades para las organizaciones.
4. Reduce los costos de gestión de identidades al eliminar intermediarios.
5. Proporciona interoperabilidad en la gestión de identidades de diferentes dispositivos.
6. Aborda el problema de la falta de mantenimiento y atención en dispositivos IoT.
7. Escala mejor que los enfoques tradicionales, especialmente en entornos IoT con una gran cantidad de dispositivos.

Finalmente, después de revisar el estado del arte sobre la aplicación de SSI al IoT, se ha descubierto que Hyperledger Indy y Hyperledger Aries son las tecnologías más avanzadas y maduras para la implementación de SSI y que la gestión de identidades para grupos de dispositivos o usuarios es particularmente compleja y no está cubierta por el estándar W3C para Credenciales Verificables (VCs). Por lo tanto, esta tesis propondrá un modelo de identidad descentralizada utilizando Hyperledger Indy y Hyperledger Aries y una solución innovadora al problema de identidad para grupos basada en SSI, denominada "Credenciales Colaborativas" que extienden el actual estándar W3C para Credenciales Verificables.

El modelo de negocio IoT-as-a-Service (IoTaaS)

El IoTaaS se concibe como un modelo de negocio en el que los propietarios de dispositivos IoT (por ejemplo, sensores de temperatura, cámaras de videovigilancia, equipos industriales, etc.) ofrecen estos dispositivos a terceros para su uso o para acceder a sus datos. Dichos consumidores pagan un precio determinado, ya sea mediante un modelo de suscripción o de pago por uso. De esta manera, se promueve la reutilización de los dispositivos existentes, evitando la proliferación descontrolada de nuevo equipamiento y potenciando la generación de ingresos para los propietarios.

El documento identifica cuatro actores principales:

- *Owner* (Propietario): entidad que posee uno o varios dispositivos IoT y los ofrece a terceros para su utilización.
- *Consumer* (Consumidor): persona o entidad interesada en consumir los datos o servicios de los dispositivos ofrecidos por el propietario.
- *Device* (Dispositivo IoT): sensor, actuador u otro equipo conectado que genera datos o provee servicios.
- *Certification Entity* (Entidad Certificadora): emite certificaciones o garantías de calidad sobre los dispositivos IoT (por ejemplo, que está calibrado correctamente, que cumple ciertos estándares de seguridad, etc.).

A menudo, la interacción se ve facilitada por un *marketplace*, que es una plataforma o interfaz que conecta a los *owners* y a los *consumers*. El *marketplace* no es estrictamente obligatorio, pero resuelve problemas de escalabilidad y coordinación, especialmente en entornos con muchos dispositivos.

Los principales desafíos tecnológicos identificados para implementar el modelo IoTaaS son:

1. **Identificación y disponibilidad de puntos de servicio**: Es necesario definir cómo se ofrecerán y accederán los servicios proporcionados por los dispositivos. El *marketplace* podría ser una solución para conectar dispositivos IoT con potenciales consumidores.

2. **Modelo de comunicación**: Se requieren protocolos seguros y eficientes para la transferencia de datos entre dispositivos IoT, consumidores y el *marketplace*. Opciones como MQTT-S o versiones seguras como SMQTT-S podrían ser apropiadas.

3. **Definición semántica del servicio**: Es crucial establecer atributos estandarizados para los dispositivos IoT, como identificador, tipo de datos, certificados de calidad, coste, ubicación y propietario.

4. **Optimización**: Se pueden explorar diversos escenarios de optimización, como la distribución óptima de dispositivos o la determinación del número ideal de certificados de calidad para maximizar beneficios.

5. **Sistema de pagos**: Es necesario implementar un sistema de micropagos eficiente y seguro, posiblemente basado en tecnología blockchain, para facilitar las transacciones entre consumidores y dispositivos o propietarios.

6. **Gestión de identidades**: Este es uno de los desafíos más críticos, ya que implica asignar identidades a los diferentes actores del sistema de manera segura y escalable. Este desafío será el que la tesis trata de resolver.

7. **Colaboración entre dispositivos**: En algunos escenarios, puede ser necesario que grupos de dispositivos colaboren para cumplir con los requisitos de un consumidor, lo que plantea desafíos adicionales en términos de gestión y coordinación. La tesis también proporciona una solución a este desafío.

La tesis también analiza múltiples casos de uso donde el IoTaaS resulta de gran utilidad. Algunos de ellos son:

1. **Investigación y recolección de datos**: por ejemplo, un investigador que necesita medir la calidad del aire en una red de sensores urbanos puede “alquilar” esos sensores. La SSI garantiza que los sensores están certificados y no han sido suplantados.

2. **Cámaras inteligentes y tráfico**: se evita la compra de equipos nuevos si ya existen cámaras disponibles con los atributos requeridos (definidos en sus credenciales). Adicionalmente, la SSI facilita la verificación de su identidad.

Gestión de identidades en IoTaaS

La gestión de identidades es uno de los problemas relevantes que se enmarca dentro del campo de la seguridad. Se puede definir informalmente como el proceso de asignar identidades a los diferentes actores en un ecosistema de TI.

Tradicionalmente, el problema de la identidad en Internet se ha abordado utilizando lo que se conoce como modelo aislado. En este modelo, el proveedor de servicios (SP) y el proveedor de identidad (IP) son la misma entidad. Los usuarios deben confiar en estos componentes para almacenar sus identidades, lo que los convierte en objetivos atractivos

para los atacantes. Con el surgimiento y el aumento de popularidad de los servicios en Internet, el modelo aislado se volvió inmanejable debido a la gran cantidad de identidades que debía gestionar mediante un solo SP, por lo que se desarrolló un modelo centralizado para resolver este problema. En este modelo, el SP y el IP no son las mismas entidades. En su lugar, diferentes SP comparten el mismo IP. Finalmente, el modelo federado difiere del enfoque centralizado en la creación de relaciones de confianza entre diferentes IP. Permite el establecimiento de redes de IP, mejorando la flexibilidad y la confianza general del sistema.

Sistema de gestión de identidades para IoTaaS

Se propone un nuevo sistema SSI para la implementación de una solución de identidad para IoTaaS, diseñado siguiendo las pautas del estándar W3C para DIDs y Credenciales Verificables (VCs). El sistema se divide en tres subprocesos:

1. Identificación e inicialización:
    1. Cada dispositivo tiene su propio DID y claves privadas/públicas almacenadas en su *wallet*.
    2. Los consumidores y entidades de certificación también generan sus propios DIDs.
    3. Se introduce un *marketplace* como solución para la identificación y disponibilidad de puntos de servicio.
2. Gestión de credenciales:
    1. Las entidades de certificación emiten VCs a los dispositivos.
    2. Los dispositivos generan Presentaciones Verificables (VPs) a partir de sus VCs y las envían al *marketplace*.
    3. Los consumidores pueden consultar el *marketplace* para verificar la autenticidad de los dispositivos.
3. Consumo de servicios:
    1. Se introduce el concepto de "credencial de consumidor" que autoriza el acceso a datos o servicios de un dispositivo IoT.
    2. El dispositivo actúa como emisor de la credencial de consumidor y como verificador cuando el consumidor quiere acceder a sus datos.

3. Se incluyen mecanismos de revocación de credenciales en caso de fallo de pago o incumplimiento de contrato.

Credenciales Colaborativas

El capítulo identifica una limitación de los esquemas SSI estándar: están concebidos, sobre todo, para identidades individuales (persona, organización o dispositivo). No obstante, en escenarios de colaboración grupal propios del IoT, como salud, logística, etc., se requiere un mecanismo que permita a un grupo de actores (humanos o dispositivos) emitir una credencial "colectiva" o "colaborativa" que pruebe la conjunción de atributos de cada integrante sin romper la privacidad individual.

Por ello, se introduce el concepto de Collaborative Credentials (CCs). Las CCs son una extensión del concepto de VCs que permite modelar la colaboración dentro de un grupo de participantes. Las CCs permiten abordar casos de uso que requieren colaboración y que no pueden resolverse con las credenciales verificables SSI estándar.

Son credenciales que:

- Se construyen de forma conjunta por los miembros del grupo.
- Permiten demostrar propiedades que requieren la contribución de varios participantes (por ejemplo, consumos agregados de un grupo de sensores).
- Permiten la colaboración entre múltiples participantes para cumplir con los requisitos de un verificador.
- Protegen la privacidad de cada parte, al exponer únicamente la información que se desea revelar.
- Permiten el desarrollo de un marco de software que cualquier verificador o titular podría usar para generar una aplicación genérica.

Aunque existen conceptos relacionados en el estado del arte, como las Credenciales de Grupo, las CCs se diferencian por su enfoque dinámico y en tiempo real. Mientras que las Credenciales de Grupo identifican a un conjunto predefinido de dispositivos o usuarios, las CCs se generan en base a la interacción activa y coordinada entre los participantes dentro de un contexto específico. Por ejemplo, en un escenario de atención sanitaria, un grupo de dispositivos médicos podría trabajar de manera conjunta para autorizar un tratamiento.

Para soportar casos de uso en los que un dispositivo actúe en nombre de otro, la tesis estudia primero la delegación de credenciales, una característica complementaria y que puede emplearse dentro del concepto de CC. Se comparan enfoques como el propuesto por la W3C (donde se "transfiere" una credencial de un actor a otro que actúa como sub-emisor) y la propuesta de Hyperledger Aries (basada en la inclusión de la VP de la VC del actor anterior dentro de la nueva credencial). Se señala que la delegación en SSI todavía presenta

limitaciones de implementación (por ejemplo, complejidad de ciertas pruebas criptográficas) y que el enfoque Aries RFC ofrece una implementación más manejable.

Modelo formal de CCs

Se define un modelo formal para las CCs, incluyendo su estructura y ciclo de vida. Los principales componentes del modelo son:

1. Participantes: Titulares de credenciales que colaboran para crear una CC.
2. Coordinador: Participante elegido para liderar el proceso de creación de la CC.
3. Verificador: Entidad que solicita y verifica la CC.
4. Emisor: Entidad que emite las credenciales individuales a los participantes.

Una CC se compone de:

1. Identificador único: Un DID o URI que identifica la CC.
2. Metadatos: Información sobre la CC, como fecha de creación y expiración.
3. Requisitos del verificador: Conjunto de condiciones que la CC debe cumplir.
4. Pruebas de los participantes: Conjunto de pruebas criptográficas aportadas por cada participante.

Asimismo, el ciclo de vida de una CC incluye las siguientes fases:

1. **Inicialización**:
    - El verificador define y publica los requisitos para la CC.
    - Los participantes potenciales descubren estos requisitos.
2. **Descubrimiento**
    - Se necesita un protocolo de descubrimiento, el cual permite al coordinador descubrir e identificar a los posibles miembros del grupo.
3. **Incorporación**
    - Una vez que el descubrimiento ha finalizado, se requiere implementar otro protocolo, denominado Incorporación.

- Este protocolo se utiliza para establecer una comunicación uno a uno entre los miembros del grupo.

- En general, la decisión de unirse o no al grupo dependerá enteramente del caso de uso, ya que a veces los miembros del grupo podrían verse obligados a unirse, gestionados por el coordinador, de modo que los miembros del grupo podrían actuar como esclavos, siguiendo las órdenes del maestro.

4. **Selección del coordinador**:

    - Los participantes eligen un coordinador mediante un algoritmo de consenso.

    - El coordinador asume la responsabilidad de recopilar y agregar las VPs.

5. **Recopilación de credenciales**:

    - Cada participante genera VPs basadas en sus credenciales individuales.

    - Las VPs se envían al coordinador de forma segura.

6. **Generación de la CC**:

    - El coordinador agrega las VPs recibidas.

    - Se crea la estructura de la CC con todos sus componentes.

7. **Presentación y verificación**:

    - El coordinador presenta la CC al verificador.

    - El verificador valida la CC comprobando las firmas y el cumplimiento de los requisitos.

Hay que tener en cuenta una limitación a la hora de realizar una implementación de SSI mediante dispositivos muy limitados, en hardware o en software. A la hora de implementar un controlador de Hyperledger Aries, es necesario que este soporte el uso de *webhooks*, de forma el agente de Hyperledger Aries pueda notificar al controlador cuando tiene que atender una petición por parte de otro agente. Algunos dispositivos pueden no soportar esto (como se verá en el apartado de implementación en el caso del Arduino SP32-C3-32S), para lo cual existe una alternativa que es el uso de *polling*. Mediante el *polling*, el controlador realiza llamadas periódicas a su agente, a fin de saber cuándo tiene una

petición que atender. Este mecanismo empeora el tiempo del protocolo, frente al uso de *webhooks*, pero en ocasiones es necesario utilizarlo.

Casos de uso

Las CCs permiten extender los casos de uso presentados anteriormente para el IoTaaS. Algunos de los nuevos casos de uso que pueden habilitar las CCs son:

1. **Dispositivos industriales y mantenimiento predictivo**: en una fábrica, diversos sensores podrían agruparse (colaborar) para presentar CCs que demuestren que la línea de producción cumple requisitos específicos de calidad.
2. **Salud y monitoreo remoto**: Un conjunto de wearables que monitorizan a pacientes podrían constituir un grupo que agrupa sus datos de forma segura y verificada.
3. **Aviación y automoción**: Se menciona la posibilidad de que vehículos con capacidad IoT intercambien información dentro de un grupo.

Evaluación de rendimiento

La evaluación de rendimiento es una parte importante de esta tesis, ya que permite validar la viabilidad y eficiencia de las soluciones propuestas. Se realizaron diversas pruebas para evaluar el rendimiento de los diferentes componentes del sistema. Las pruebas son las siguientes:

Evaluación del modelo de identidad para IoTaaS

Se implementó un prototipo del sistema de gestión de identidades propuesto para IoTaaS utilizando Hyperledger Aries. Las pruebas se realizaron en un entorno controlado con un dispositivo IoT simulado mediante una Raspberry Pi y un consumidor en un ordenador Dell Latitude 5580 equipado con un procesador Intel® Core™ i7-7600U y 8GM de memoria RAM. Los resultados mostraron que:

- El tiempo promedio para la creación de VCs fue de 0.03 segundos por dispositivo.
- El intercambio de 10000 credenciales de consumidor tomó en promedio 0.8 segundos.
- La verificación de 10000 presentaciones verificables requirió 0.9 segundos en promedio.

Estos tiempos de respuesta son más que aceptables para la mayoría de los escenarios de IoTaaS, demostrando la viabilidad del modelo propuesto.

Evaluación de la delegación de credenciales

Se ha realizado una prueba con el mecanismo de delegación de credenciales propuesto por Hyperledger Aries para comprobar cuánta sobrecarga añade al flujo básico de intercambio de credenciales de SSI. Esta prueba se ha realizado mediante contenedores Docker instalados en el equipo Dell presentado anteriormente. El cliente utilizado para la prueba ha sido Hyperledger Indy. Los resultados muestran que, en media, añade una sobrecarga de 0.24 segundos al flujo normal de SSI, por lo que el mecanismo de delegación propuesto no supone una sobrecarga elevada.

Evaluación de las Credenciales Colaborativas

Se implementó un prototipo de CCs y se evaluó su rendimiento en diferentes escenarios:

- Se probaron grupos de 5, 10, 20 y 50 participantes.
- Se midieron los tiempos para la creación de CCs, verificación y revocación.

Los resultados mostraron que:

- El protocolo propuesto no implica un empeoramiento en términos de rendimiento frente a un flujo normal de Hyperledger Aries.
- El protocolo se ve más afectado por la potencia computacional de los miembros del grupo que por los pasos del protocolo en sí.
- En cuanto al gasto energético, en una batería normal de 5200 mAh, dicha batería instalada en uno de los participantes ejecutando constantemente el protocolo duraría alrededor de 11,36 días sin carga, lo cual es aceptable en un entorno como el considerado.
- El tiempo total no solo no aumenta exponencialmente cuando el número de miembros del grupo aumenta linealmente, sino que aumenta más despacio que linealmente. El experimento realizado también ha demostrado que las fases de descubrimiento e incorporación son las más afectadas con un aumento del número de miembros del grupo. Esto revalida el argumento de que es deseable minimizar la ocurrencia de las fases de descubrimiento e incorporación.
- Probando el protocolo con y sin el coordinador, se observa que el uso de un coordinador no aumenta dramáticamente el tiempo del protocolo, mientras que aporta muchas ventajas al mismo, estando así justificada su inclusión.

Implementación de un controlador de Hyperledger Aries en un dispositivo Arduino

Mediante este experimento se pretende demostrar que es posible para un dispositivo muy restringido (Arduino SP32-C3-32S) correr un controlador de Hyperledger Aries. Los resultados muestran que un controlador de Hyperledger Aries puede implementarse sin ningún problema. Sin embargo, es necesario realizar *polling*, ya que un Arduino no tiene la capacidad de implementar *webhooks* HTTP.

Se concluye por tanto que las propuestas en esta tesis son viables y eficientes para su implementación en escenarios reales de IoTaaS. El sistema de gestión de identidades basado en SSI, la delegación de credenciales utilizando el enfoque Aries RFC y las Credenciales Colaborativas ofrecen un buen equilibrio entre seguridad, privacidad y rendimiento.

La escalabilidad demostrada en las pruebas sugiere que estas soluciones pueden adaptarse a la creciente demanda de dispositivos IoT y servicios relacionados. Sin embargo, se recomienda realizar pruebas adicionales en entornos de producción a gran escala para validar completamente el rendimiento en condiciones reales de uso.

Conclusiones

Según las conclusiones proporcionadas en cada capítulo, podemos establecer las siguientes conclusiones generales de esta tesis:

Formalización de IoTaaS y beneficios en sostenibilidad.

El concepto de IoTaaS, aunque ya ha sido discutido en la industria y el ámbito académico, aún no ha sido formalmente definido ni formalizado. Esta tesis aborda esta carencia proporcionando una formalización integral del modelo de negocio IoTaaS, identificando roles, interacciones y flujos operativos. El análisis subraya el potencial de IoTaaS para contribuir significativamente hacia objetivos de sostenibilidad. Al promover la reutilización de recursos IoT existentes y facilitar el acceso bajo demanda a dispositivos IoT, IoTaaS minimiza el impacto ambiental relacionado con la producción de nuevos dispositivos y maximiza la utilización del hardware ya desplegado. En consecuencia, fomenta prácticas como reutilización, reutilización con nuevo propósito y reciclaje, alineándose con las metas globales de sostenibilidad y reduciendo los costos económicos y ambientales generales.

Retos tecnológicos del modelo IoTaaS e importancia de la gestión de identidades.

La implementación del modelo IoTaaS presenta diversos desafíos tecnológicos, especialmente relacionados con seguridad, estándares de comunicación, definiciones semánticas, mecanismos de pago, procesos de optimización y colaboración entre

dispositivos. Dentro de la seguridad, la gestión de la identidad ha surgido como un desafío fundamental que afecta de manera crítica a la seguridad en IoTaaS. Debido a la gran cantidad de dispositivos interconectados y a la necesidad de procedimientos robustos de autenticación y autorización, la gestión precisa de las identidades de los dispositivos es esencial. En particular, la tesis destaca que las soluciones tradicionales centralizadas de gestión de identidad no alcanzan la escalabilidad y seguridad requeridas. En este contexto, la SSI emerge como un paradigma prometedor para abordar los desafíos de gestión de identidad en entornos IoT, ofreciendo ventajas frente a métodos tradicionales, particularmente en seguridad, privacidad e integridad de datos en ecosistemas IoT. Se introduce además el concepto de CCs como una nueva solución para gestionar identidades en grupo en escenarios IoT. Las CC abordan situaciones que requieren colaboración entre múltiples entidades, casos que no se resuelven plenamente con implementaciones estándar de SSI tradicional. El modelo formal para CC, incluyendo su estructura y ciclo de vida, proporciona una base para implementar la gestión de identidad basada en grupos en IoT.

Aplicación de SSI a IoT e IoTaaS: viabilidad y limitaciones.

La investigación demostró exitosamente que los esquemas de SSI pueden solucionar el problema de gestión de identidad en entornos IoT. Usando DIDs y VCs, las soluciones SSI mejoran significativamente la privacidad, la escalabilidad y la interoperabilidad de dispositivos en los escenarios IoTaaS. En particular, la aplicación de SSI en IoTaaS mitiga desafíos comunes de IoT, tales como interdependencia, diversidad o falta de atención, entre otros. Sin embargo, a pesar de estas ventajas, la tesis también identificó limitaciones específicas al implementar SSI en dispositivos IoT, especialmente aquellos con recursos limitados. Estas limitaciones incluyen sobrecarga computacional en operaciones criptográficas requeridas por SSI (por ejemplo, generación de pruebas de conocimiento cero), o la necesidad de enfoques híbridos o basados en proxy en dispositivos extremadamente limitados. Tales limitaciones apuntan a la necesidad de optimización adicional o adaptación de protocolos SSI para garantizar su aplicabilidad general en distintos contextos IoT.

Mecanismos de delegación y adaptaciones necesarias en SSI.

Como preámbulo al problema de CC, la tesis investigó exhaustivamente mecanismos de delegación dentro de SSI, esenciales en escenarios donde las credenciales de identidad deben ser delegadas o reenviadas entre entidades. Las tecnologías actuales tienen deficiencias significativas relacionadas con la complejidad en la emisión y verificación de credenciales, así como en la interoperabilidad entre diferentes tecnologías de SSI. Un análisis detallado identificó dos propuestas de delegación principales: el estándar del W3C y el RFC de Hyperledger Aries. La especificación W3C mostró falta de soporte en algunas tecnologías SSI existentes y complejidad de implementación, puesto que exige almacenar en la *wallet* del titular VCs cuyo sujeto no es él mismo. Dicha limitación no está presente

en la propuesta Aries RFC, que almacena VPs embebidas en lugar de VCs. Por ello, el mecanismo del Aries RFC es más compatible con las tecnologías SSI existentes, aunque presenta también sus propios inconvenientes (como ciertos mecanismos basados en números de secuencia). Esto pone de manifiesto que las tecnologías actuales no siempre cumplen completamente con el estándar W3C para VCs, existiendo aún una brecha considerable entre el estándar y sus implementaciones. Estos hallazgos subrayan claramente la necesidad de adaptación y estandarización de mecanismos de delegación para explotar plenamente el potencial de SSI.

Introducción y aplicabilidad de las Credenciales Colaborativas (CC).

Las CC, tratadas en esta tesis, representan una extensión novedosa a esquemas tradicionales de SSI, permitiendo la utilización de credenciales mediante colaboración en tiempo real entre múltiples entidades. Los esquemas estándar SSI actuales, definidos típicamente por los estándares del W3C de DID y VC, se enfocan principalmente en gestionar identidades individuales. Sin embargo, este enfoque tiene limitaciones en escenarios que requieren colaboración en tiempo real entre varias entidades con credenciales diferentes.

Las CC atienden eficazmente casos donde las soluciones tradicionales SSI son insuficientes. Por ejemplo, escenarios en salud, cadena de suministro e Industria 4.0 se benefician enormemente de las CC, incluyendo el caso de uso del IoTaaS. Los ecosistemas IoT requieren inherentemente interacción y colaboración entre diversos dispositivos. Las CC permiten a grupos de dispositivos IoT o proveedores de servicios agregar dinámicamente sus credenciales, proporcionando una autenticación colectiva que simplifica flujos operativos y robustez frente a brechas de seguridad. Formalmente, las CC permiten a múltiples titulares de credenciales colaborar dinámicamente, satisfaciendo colectivamente los requisitos del verificador y ampliando así la aplicabilidad de SSI hacia escenarios previamente no cubiertos.

Viabilidad y desempeño de las CC en dispositivos IoT.

Finalmente, el Capítulo 6 proporcionó validación empírica confirmando que las implementaciones SSI y CC son viables y prácticas en dispositivos IoT reales. La evaluación de rendimiento demostró que dispositivos IoT, aun con recursos computacionales limitados, pueden implementar efectivamente esquemas SSI sin comprometer el rendimiento. La experimentación indicó que los mecanismos de delegación y procesos de manejo de CC pueden realizarse dentro de tiempos aceptables, confirmando su aptitud para ser desplegados en escenarios IoT realistas. Las pruebas mostraron que las operaciones relacionadas con CC pueden optimizarse adecuadamente y ejecutarse con eficiencia. Esta tesis valida concluyentemente la viabilidad práctica y apoya la integración de conceptos SSI y CC en IoTaaS y en ecosistemas IoT más amplios.

Trabajo futuro.

Las siguientes líneas de investigación podrían servir como trabajo futuro en el campo de IoTaaS y CCs: a) optimización de la escalabilidad: análisis más profundo de algoritmos de consenso ligeros y cadenas laterales para soportar tasas de transacción más altas; b) modelos económicos y de precios: incorporación de funciones de oferta-demanda en el mercado, incentivos para la calidad y cooperación entre propietarios de dispositivos; c) privacidad mejorada: aplicación de técnicas avanzadas de criptografía (zk-SNARKs, cifrado homomórfico, etc.) para garantizar niveles más altos de confidencialidad; d) soporte para grupos grandes: expansión de CCs para escenarios que involucren miles de dispositivos y estudio de protocolos de coordinación sin un coordinador central; e) integración con, o inclusión en, estándares internacionales: cooperación con consorcios como W3C o ESSIF para unificar especificaciones para el uso de SSI y CCs en dispositivos IoT.

This page was intentionally left blank.