UNIVERSIDADE D
COIMBRA

Kevin Martins Carvalho

PRIVACY AND LOCATION IN IOT

Dissertation in the context of the Master in Informatics Engineering, Specialization in Communications, Services and Infrastructures, advised by Prof. Jorge Granjal and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

June 2021

Acknowledgements

Firstly, I would extend my sincere gratitude to Dr Jorge Granjal for all the guidance throughout every stage of this thesis. His relentless help and knowledge were key factors in the elaboration of this work.

I would also like to thank my family and girlfriend for their support during all these months, especially at this moment in human history that we are facing. I am grateful to all my friends and university colleagues for all the direct and indirect support.

I sincerely appreciate all the work that the academic community performs to formulate new ideas and solutions to improve our society.

Last but not least, a big salute to the open-source and blockchain community for all the tremendous work they provide to society.

Resumo

A privacidade dos dados e da localização em aplicações da Internet of Things (IoT) não tem sido suficientemente abordada, visto que não acompanha o ritmo de desenvolvimento desta indústria. O que resulta na falta de mecanismos de privacidade, tornando estas aplicações vulneráveis a uma série de ataques. Portanto, as aplicações IoT devem implementar mecanismos de privacidade por omissão, dado que estas aplicações podem gerar uma grande diversidade de dados, compreendida entre dados de telemetria e informação sensível dos utilizadores. Como muitas destas aplicações lidam com informação sensível do utilizador, devem ter em conta o Regulamento Geral sobre a Proteção de Dados (RGPD).

Nesta tese, é feito um estudo do estado da arte sobre privacidade na IoT, baseado em artigos e inquéritos que investigam este tópico e propõem soluções inovadoras para mitigar esta questão. Uma das soluções inovadoras identificadas foi o uso da tecnologia *blockchain* em aplicações IoT para melhorar a sua privacidade e segurança. Além disso, estas propostas foram analisadas com base nos seus mecanismos de privacidade e segurança, e com base numa *framework* identificada num inquérito. Algumas limitações e riscos também foram diagnosticas. Esta análise realizada serviu como base para a formulação da nossa proposta de preservação de privacidade focada em aplicações IoT com mobilidade (e.g., veículos inteligentes).

Para além de mecanismos de privacidade, um dos objetivos principais da nossa solução é implementar mecanismos que permitam o utilizador ter um maior controlo sobre a exposição dos seus dados. Além disso, várias tecnologias são implementadas com o fim de tirar partido dos benefícios de cada uma e mitigar algumas das suas limitações. As quais são: *blockchain*, armazenamento descentralizado e o Message Queue Telemetry Transport (MQTT). Por fim, fazemos uma análise da nossa proposta através da validação dos requisitos e através da comparação entre as propostas analisadas anteriormente.

Palavras-Chave

Privacidade, IoT, localização, segurança, blockchain, MQTT

Abstract

Privacy of data and location in IoT applications has not been addressed sufficiently since it has not kept up with the pace of development of this industry. It results in a lack of privacy mechanisms, making these applications vulnerable to various attacks and exploits. Therefore, IoT applications must implement it by design due to the high diversity of data generated, ranging from telemetry data to users' sensitive data. Given that most of these applications handle user sensitive information, General Data Protection Regulation (GDPR) should be taken into consideration.

In this thesis, a state-of-the-art study on privacy in IoT is provided, based on papers and surveys that research this topic and propose innovative solutions to mitigate this issue. One of these innovative solutions is blockchain technology used in IoT applications to enhance its privacy and security. These proposals were analysed based on their privacy and security mechanisms and based on a framework identified in the survey. In addition, some of their limitations and vulnerabilities were also detected. This analysis was used as a foundation for formulating our privacy-preserving proposal focused on mobile IoT applications (e.g., smart vehicles).

Besides of privacy mechanisms, one of the main focus of our solution is also to providing user-centricity, in which the user has more control over his data exposure. Furthermore, it implements various technologies to take advantage of their benefits and mitigate some of their limitations. Some of which are: blockchain, decentralised storage and Message Queuing Telemetry Transport (MQTT). Lastly, we provide an analysis over our proposal, by validating the formulated requirements and comparing it to the previously analysed proposals.

Keywords

Privacy, IoT, location, security, blockchain, MQTT

Contents

Acronyms

ABE Attribute Based Encryption. 27, 32

ACL Access Control List. 27, 30, 32

AES Advanced Encryption Standard. 27, 57

AES-256 Advanced Encryption Standard 256-bit key. 56

AML Anti-Money Laundering. 21

AMQP Advanced Message Queuing Protocol. 7

CoAP Constrained Application Protocol. 7

DoS Denial-of-Service. 1, 21, 40, 47

DTLS Datagram Transport Layer Security. 10

ECC Elliptic-curve cryptography. 27, 32

ECDSA Elliptic Curve Digital Signature Algorithm. 17, 57

ECIES Elliptic Curve Integrated Encryption Scheme. 57

EU European Union. 1

GDPR General Data Protection Regulation. vii, 1

GPS Global Positioning System. 8

IoT Internet of Things. v, vii, 1, 2

IPFS Interplanetary File System. 2, 29, 30, 32, 33

kB Kilobyte. 60

kbps Kilobits per second. 60

KYC Know Your Costumer. 21

M2M Machine to Machine. 21

MQTT Message Queue Telemetry Transport. v, 7, 11, 24

ms Milliseconds. 27, 60

OS Operating System. 44, 57

OTP One-Time Password. 30, 32

PbD Privacy by Design. 25, 26

PBFT Practical Byzantine Fault Tolerance. 19

PBVU Private Blockchain of Vehicle Users. 39

PET Privacy-Enhancing Technology. 25, 26, 31, 33, 37, 39

PoB Proof of Burn. 18

PoS Proof of Stake. 18

PoW Proof of Work. 18, 19

QoS Quality of Service. 10, 12, 13, 24

RSA Rivest-Shamir-Adleman. 17

SASL Simple Authentication and Security Layer. 10

SIoV Social Internet of Vehicles. 22

SNMP Simple Network Management Protocol. 14

TCP Transmission Control Protocol. 9

TLS Transport Layer Security. 10, 15, 48

UDP User Datagram Protocol. 10

UI User Interface. 57

VM Virtual Machine. 5, 56, 57

XML Extensible Markup Language. 10

XMPP Extensible Messaging and Presence Protocol. 7

List of Figures

List of Tables

Chapter 1

Introduction

Internet of Things (IoT), nowadays, is a concept that covers a broad spectrum of areas and scenarios of applicability. Its primary purpose is to connect objects capable of collecting data (e.g., temperature sensors) to the internet or a local network to automate specific tasks or telemetry analysis. The applications can extend from simple networks (e.g., smart homes) to complex and intricate networks (e.g., industrial agriculture and automotive industries). This broad spectrum of applications has contributed to the exponential growth of IoT devices connected to the network. According to the International Data Corporation (IDC), it is estimated that 55.7 billion IoT devices will be connected worldwide, generating 73.1 zettabytes (ZB) of data in 2025 [44].

This growth in implementation means that the adoption of the technology is spreading worldwide and is also starting to be used in complex applications, such as smart cars, smart cities, and industry 4.0. However, this amount of diversified data collected and shared by IoT devices brings up some concerns about privacy and the users' awareness about how their data is processed. Privacy of data is essential, whether it is sensitive information about users or their devices. It is a matter of right and freedom to have privacy and control our data. If it is not safeguarded, the users' identity and personal information can be threatened by a series of potential attacks, e.g., user profiling and user tracking. In recent years, users data has been endangered and used by various corporations for user profiling and user tracking. Such as the Cambridge Analytica case [60], which was a company that used the private information of 87 million Facebook users without their consent to gather information about their political preferences and other personal information. Some speculations claim that this information was used to manipulate elections.

The General Data Protection Regulation (GDPR) [26], which was launched on May 25th of 2018 by the European Union (EU), has the goal to give control to EU citizens over their data. Since IoT applications have been extended to the users' private daily life, a high amount of private data is gathered. Thus, these applications must implement user-controlled privacy and privacy solutions to comply with GDPR.

Current storage solutions (e.g., centralised cloud storage) used in some IoT applications do not comply with the security and privacy requirements of these types of applications. Since the privacy of the data is not guaranteed, the user cannot control who can access it, and they are vulnerable to certain failures (single points of failure) and attacks (Denial-of-Service (DoS)). However, blockchain is a solution that intends to solve the risks inherent in centralised systems and provide privacy to users by design. It is a decentralised, distributed, and immutable ledger currently being adopted and developed to be integrated into various applications. Some of these applications are: decentralised voting system to

prevent election fraud [72], decentralised financial systems [27] or decentralised storage [64]. Integrating blockchain with IoT applications can be used to enhance the security and privacy of users and their data.

Some proposals present in the literature use blockchain to enhance the privacy and security of telemetry data and sensitive data generated in IoT applications. However, the data is stored in external systems that are centralised or do not guarantee the removal of data (e.g., Interplanetary File System (IPFS)) or are publicly stored in the public blockchain. Based on those proposals, this thesis proposes an architecture for IoT applications with mobility (e.g., smart vehicles) since the collected data can contain geolocation information, which is sensitive data since the user could be tracked if no privacy measures are guaranteed. Additionally, the proposal intends to solve some limitations and privacy problems identified in the state-of-the-art and provide a more GDPR-compliant system.

The main objectives of this thesis are to provide an in-depth study of the privacy of data and location in IoT environments and formulate a privacy proposal based on the state-of-the-art study. The objective of the thesis can be further divided into:

- Provide an overview of the transmission protocols used in IoT. One of the protocols is elected to be further analysed and implemented in our proposal, based on its characteristics.
- An overview of the blockchain technology, such as its functionalities, applicability in IoT and major vulnerabilities.
- Provide a state-of-the-art on the concept of privacy-focused in IoT, where its main characteristics and Privacy-Enhancing Technologies (PET) are identified and analysed. Our proposal is based on the identified proposals, although it will provide some distinct features.
- Propose an IoT privacy-enhancing proposal architecture using blockchain and MQTT. Its applicability is described, its requirements, architecture elements, implementation and evaluation strategies, and all underlying technologies used to implement and test this system.
- The proposed system will be implemented and tested in a controlled environment to validate the formulated requirements, and assess its usability in a real environment and identify further work to be done.

Chapter 2

Schedule and methodology

This chapter provides an overview of all the done tasks to accomplish this thesis and their respective planned schedule. We will compare the actual executed and planned schedule and identify mishaps encountered to reflect on the performed work. Finally, the methodology used in the research phase of this document is presented.

2.1 Planned schedule

The Gantt chart of Figure 2.2 is represented the schedule and time estimation of each phase of the thesis.

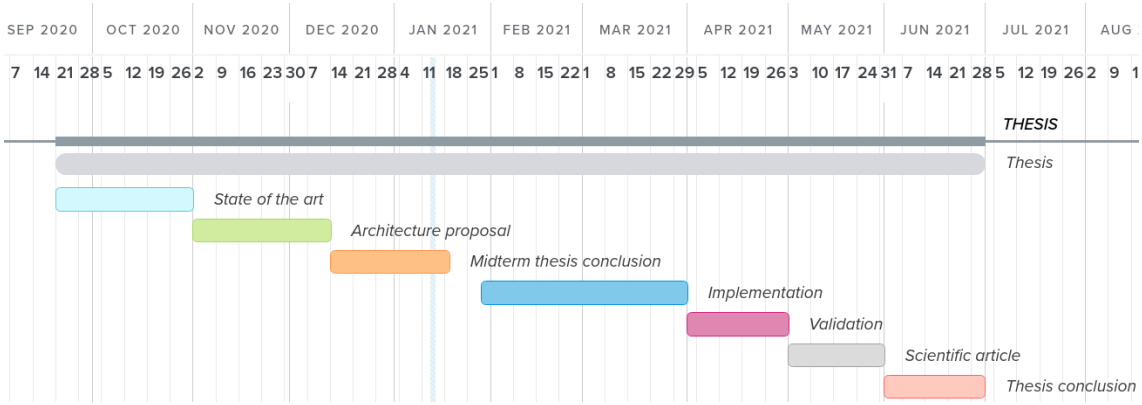

Figure 2.1: Gantt chart with the planning and estimation of each thesis phase.

- **State-of-the-art** (21/09/2020 to 30/10/2020), a study of the current state-of-the-art of privacy in IoT. Analysis of papers and surveys related to privacy in IoT and privacy-enhancing proposals.
- **Architecture proposal** (2/11/2020 to 12/11/2020), privacy-enhancing proposal, focused on IoT, based on the state-of-the-art previously done.
- **Midterm thesis conclusion** (14/09/2020 until 18/01/2021), the conclusion of the midterm thesis, which is composed of the contextualisation of the main technologies being studied, state-of-the-art of privacy in IoT and the architecture of the privacy-enhancing proposal of this thesis.

- **Implementation** (28/01/2021 to 31/03/2021), implementation of the main components of the proposal.
- **Experiments and validation** (01/04/2021 to 30/04/2021), tests to validate the requirements of the proposal.
- **Scientific article** (03/05/2021 to 31/05/2021), writing of a scientific article related to the proposal, identifying its purpose, implementation, and outcomes.
- **Thesis conclusions** (01/06/2021 to 30/06/2021), the conclusion of the thesis, which consolidates all the work done throughout the academic year.

2.2 Accomplished schedule

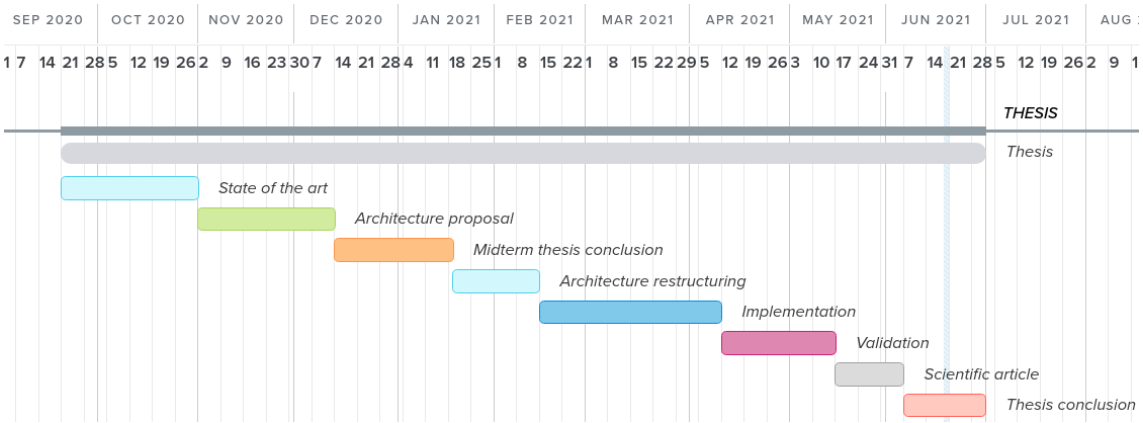

Figure 2.2: Gantt chart with the actual executed schedule.

The first phase of this thesis was concluded within schedule, which was the state-of-the-art architecture proposal and midterm thesis conclusion. However, some modifications were implemented in the proposal during the second phase, which delayed its implementation. These modifications were related to GDPR compliance since the initial version did not address this subject. Therefore, more research was done to consider privacy aspects that GDPR defines and integrate them in the architecture and validate them.

During the experiments and validation stage, some problems related to the synchronisation of the blockchain nodes with public blockchain also affected this phase. The problems were related to the hardware resources. Namely, the read/write speed of hard drive disks (HDD), which was a significant bottleneck for this process. Thus, to speed up this process and start testing the system as soon as possible, a solid-state drive (SSD) was used, accelerating the process from 1 week to 1-2 days.

All these mishaps delayed some stages of this thesis. However, we managed to implement and test our proposal as planned.

2.3 Methodology

In order to perform a relevant study of the state-of-the-art oriented to privacy in IoT, a methodology was defined. A list of restrictions was defined to reduce the volume of work to research and focus on more relevant work.

The main factor was restricting the research to specific key terms, such as "Privacy in IoT", "Privacy proposals for IoT", "Location and data privacy in IoT", and "User privacy in IoT". In order to obtain more updated information, the analysed papers were at a maximum of six years old, thus, from 2014 to 2020 (considering that this research was initiated in 2020).

Initially, some research was performed to assess the privacy threats, challenges, and requirements of IoT applications. Therefore, some surveys [74, 81] and papers [68, 86, 87] were reviewed. Through this inquiry, some base ground was created for the following research phases.

At first, MQTT was defined as the main technology to study due to our previous knowledge and practical experience. Similarly to the previous approach, a set of surveys [53, 61, 85] were analysed to obtain more detailed information about the technology and how it compares to other technologies. Afterwards, privacy proposals using MQTT [56, 65, 78, 84] were examined. However, the paper [56] introduced the concept of integrating blockchain in a system to provide more privacy and security to the user, which changed our technological approach.

A set of surveys [34, 58, 66, 82] focused on blockchain were studied to gather information about its security and privacy functionalities and possible integrations with IoT applications. Therefore, some privacy proposals that used blockchain were analysed [49, 62, 63, 76, 95]. These proposals and the previously identified MQTT based proposals were analysed and classified based on their privacy mechanisms, technologies and approaches, which allowed to identify certain limitations and improvements. Our proposal was formulated based on this analysis.

During the proposal's development phase, several tests were performed to identify possible bugs and technical problems before implementing and performing experiments. The system's components were all integrated into the same host as Virtual Machine (VM) to use the same hardware characteristics since using different machines could influence the results. In addition, the hardware resources were distributed heterogeneously among the VMs, considering that each component has a different computational requirement. In the experiments, the VMs that did not participate in the specific scenarios or requests were shut down to maximise the resources available to the remaining VMs.

In general, the whole process of this work was iterative. Its aspects were not all defined *a priori*. The only predefined goal was to develop a privacy-enhancing proposal for IoT applications.

Chapter 3

Background

In this chapter, privacy-related aspects and requirements of IoT applications will be introduced, as well as some data transmission protocols used in IoT (Advanced Message Queuing Protocol (AMQP), Constrained Application Protocol (CoAP), Message Queue Telemetry Transport (MQTT) [61, 79], and Extensible Messaging and Presence Protocol (XMPP)). Addressing these protocols explains why the MQTT was chosen to be implemented in the privacy solution of this thesis, focused on IoT applications. Then, a more detailed description of the MQTT protocol is made.

In addition, contextualisation of blockchain technology will also be provided. The main characteristics will be addressed and the benefits it brings for IoT applications regarding data privacy and security. Moreover, its main vulnerabilities and limitations are identified.

3.1 Privacy in Internet of Things (IoT)

The number of devices and IoT applications has increased exponentially in recent years. However, IoT's security and privacy area has not kept up with its growth pace, derived from the implicit challenges to achieve a balance between security quality and performance. One of the factors is the heterogeneous nature of its networks.

Despite being a key aspect in IoT applications and solutions, privacy is a topic that is not addressed enough. Its implementation is critical for protecting some types of data, e.g., sensitive user data, geolocation data, and telemetry data. Traffic analysis attacks, such as eavesdropping, can analyse the data transmitted on the network and capture it. Unauthorised access to MQTT topics could compromise the data being transmitted by devices to the topics, threatening the integrity of the data.

By all means, it is essential to take into account privacy when designing an application that handles sensitive data (Privacy by Design [81]). For example: in application-related health care [87], mechanisms are needed to protect sensitive user data, such as location and health history. Moreover, in applications related to Smart Parking ([68, 86]), it is important to protect the location of the vehicles, their telemetry data, and their owners' sensitive data.

The key aspects of privacy in IoT applications are:

- **Data privacy**, it consists of data generated (e.g., telemetry data). In the case of Smart Homes [81], thieves can obtain information related to the periods of absence

of users in their home through data generated by power switches or appliances (e.g., they stay in the "off" state at a certain time and reconnect after a few hours).

- **Location privacy**, the privacy of data related to the location of an object or person (considered sensitive data). It can be Global Positioning System (GPS) data [86], or the spatial location within a building or a certain area [68]. Therefore, it is important to provide privacy for this type of data to prevent user tracking.

- **User privacy**, the sensitive user data that is transported over the network and stored in an external facility (e.g. cloud storage), can disclose the identity of the user if privacy-enhancing measures are not integrated. This exposure may result in user tracking (e.g., obtaining a person's daily activity history) and user profiling (data can be used without the user's consent by companies).

- **User-controlled privacy**, the user has the power to define his privacy preferences and has more control over his data. In other words, the user can define the data's level of exposure, which data he wants to share and to whom, and also limit its usage.

The perfect privacy solution does not exist because many aspects restrict the quality of the solution. Especially in IoT, the main constraints are computational power and energy efficiency. Therefore, privacy solutions for these applications have to be well-balanced in terms of computational resources and complexity, which also applies to security.

3.2 Data Protection

Some areas of the IoT industry work with personal data. Thus, these applications must implement strategies to provide protection to the users' data and comply with GDPR [26]. GDPR is a regulation that provides fundamental directions in order to accomplish an equitable treatment of the third parties and EU users [74]. In essence, it intends to give the users control over their data to provide privacy rights to the user, such as the right to be forgotten (delete data), informed, and restrict processing.

In the survey [74], the authors provide an analysis of GDPR focused on IoT. They analysed 29 papers that proposed IoT based privacy-preserving proposals, which were evaluated according to the 19 formulated GDPR essential characteristics. These characteristics were mapped to the four challenges proposed in [91], which are: "Profiling, inference and discrimination" (prevent user profiling, data inferences and discrimination based on the user's data), "Control and context-sensitive sharing" (provide control to the users over their data), "Consent and uncertainty" (user-controlled privacy) and "Honesty, trust, and transparency" (trust relationship between devices, third parties and users). Through the performed analysis, they verified that machine learning was the most used technique to improve privacy, such as automating privacy preferences and computing risk probabilities. However, characteristics such as data erasure, prevent excessive data collection and other transparency-related characteristics were the least addressed features.

In the following table, the GDPR required characteristics identified in [74] are enumerated and described. These characteristics are useful to evaluate privacy proposals in terms of their compliance with GDPR.

Table 3.1: GDPR required characteristics identified in [74].

GDPR characteristic	Description
CR1-Prevent inference	Prevent the processing of personal data that can reveal more information about a person, such as ethnic origin, political opinions, or even the possibility to uniquely identify a person (Article 9 of GDPR [26]).
CR2-Provide data transformation	Techniques to provide privacy to users' data, prevent data inference and profiling attacks (e.g., data anonymisation) (Recital 26).
CR3-Provide user awareness on data collection	Provide data collection awareness to the user (Article 12).
CR4-Provide control of personal data to users	The user can specify his privacy preferences over his data.
CR5-Provide monitoring and control of devices that collect data	Users can control and monitor data collection, specifying who can access it and what actions can be made.
CR6-Provide tools for data management to users	Tools that allow the user to manage and control usage over their data.
CR7-Provide ability for data erasure	Provide the ability to delete or rectify data (Article 17).
CR8-Provide transparency	Offer awareness to the user about how their data is processed.
CR9-Provide balance of privacy between the user and third parties	Inform the user about the possible risks of sharing his data to decide whether to take these risks in exchange for potential benefits.
CR10-Provide enforcement of user privacy preferences	Enforce the user's privacy policies.
CR11-Provide privacy by design or privacy by default	E.g., implement privacy throughout the whole process of developing a product or enforce privacy policies by default (without the need for user intervention) (Article 25).
CR12-Provide ability to users to make informed consent choices	Inform the user about possible privacy risks when performing certain choices (Article 7).
CR13-Estimate privacy risks of data collection/inference to users	Compute the privacy risk level based on the user's privacy preferences or other factors.
CR14-Communicate risks of data collection/inference to users	Inform the user about possible privacy risks,e.g., in scenarios where he uses more vulnerable privacy preferences or provides data to an organisation with a bad reputation.
CR15-Provide ability to users to specify their privacy preferences	The user can impose policies to determine how his IoT devices and data can be used.
CR16-Prevent excessive data collection	Implement mechanisms to minimise data collection, which provides the required amount to complete a task successfully.

3.3 Data transmission protocols

Multiple data transmission protocols can be used in IoT applications, including AMQP, CoAP, MQTT, and e XMPP. All of them have their advantages and disadvantages, which differentiates their end application.

- **AMQP** is an *open standard* protocol that uses Transmission Control Protocol (TCP) in its transport layer, which is a more reliable protocol to perform communications. It allows interoperability between heterogeneous applications and systems (an important aspect for IoT) [61]. This protocol is quite complete (compared to other protocols), some of its features are:
  - It allows several approaches in the communication between the exchange (a component that receives the messages from the *publishers*) and queues inside the broker (AMQP server).

  - Two approaches can be implemented in the communication between queues and customers: messages sent to customers subscribed to the queue (asynchronous), or customers can get the messages on demand (synchronous).
  - It has three levels of Quality of Service (QoS).
  - Flow control that allows traffic shaping to be done if there is congestion on the network.
  - Security mechanisms, such as Transport Layer Security (TLS) and Simple Authentication and Security Layer (SASL) (a framework used to implement authentication mechanisms in application layer protocols).

  Although it is a feature-rich protocol, it requires some computational power and bandwidth. In [71], it is verified that AMQP has low reliability in networks with low bandwidth, which is not desirable for constrained environments.

  Initially, it was used for messaging in business environments. It was developed as a non-proprietary solution that managed many message exchanges that could be exchanged in a short period in the system. The protocol fits in environments with no concern about resource usage (e.g., communication between servers in the cloud).

- **CoAP** has a synchronous communication model (request/response). It uses the User Datagram Protocol (UDP) protocol for packet transportation, which makes it a more lightweight protocol (smaller packets). However, it also reduces its reliability on unstable communication channels, and some problems, such as packet loss and packet disorder, can occur. To get around this reliability problem, CoAP has message types that ensure the delivery of packets: *Confirmable* and *Acknowledge*.

  It supports Representational state transfer (REST) type architectures by using Hypertext Transfer Protocol (HTTP) methods (GET, POST, PUT and DELETE), which allows devices to communicate directly with web servers. To ensure some level of security, CoAP uses Datagram Transport Layer Security (DTLS), a TLS-based protocol that provides integrity and confidentiality to UDP communication. However, DTLS is not designed for devices with limited resources because it requires many computational resources.

  Despite lacking adequate security mechanisms for IoT applications, this protocol can be used for message transmission between devices with few resources.

- **MQTT** is an asynchronous message transmitting protocol with a *publish/subscribe* model. It uses a broker (MQTT server) that manages the topics, where messages are allocated, identified, and transmitted to MQTT subscribers of the topics. It uses TCP to transport the messages, which offers greater reliability to the protocol. It also provides resilience mechanisms (*wills* and *durable connections* - will be covered in section 3.4) and QoS with three levels. It has some security mechanisms, such as TLS, authentication and user identification. However, like the other protocols, these security mechanisms were not developed to be used by constrained devices.

  Compared to CoAP, MQTT is also a lightweight protocol, making it ideal for applications with limited resources. Due to its resilience mechanisms, it is feasible for applications with unstable communication channels (applications with mobility) and with low bandwidth (applications that use low-frequency channels to communicate over long distances, such as LoRaWAN [41] and SigFox [46])

- **XMPP** is an open standard protocol, which supports synchronous (request/response) and asynchronous (*publish/subscribe*) communications and uses TCP. Messages are text-based using Extensible Markup Language (XML) format. It supports various

types of connection technologies, for instance, multimedia sessions (voice chat, video chat, and file transfer) and HTTP connections (synchronous communication). It has an addressing mechanism for each device on the network, which other protocols do not have implemented. It also supports TLS and SASL to introduce security into the system's architecture.

This protocol was initially designed for instant messaging between applications using XML. Due to the size of the XML packets, it is not feasible for networks with limited bandwidth. The absence of resiliency mechanisms and QoS is another factor that makes this protocol unsuitable for applications with unstable connections and limited resources. This protocol is more suitable in real-time applications (multimedia sessions, instant messaging) and some IoT applications without limited resources.

These application layer protocols offer beneficial features for IoT applications and other applications. A summary of their characteristics is present in the following table. However, due to the complexity of some of them, they are not feasible for certain applications where computational resources and network infrastructure are scarce. MQTT has been growing in use in IoT applications ([61]) due to its lightweight nature, resilience and QoS mechanisms. However, it lacks adequate privacy and security mechanisms for IoT. These aspects make MQTT an interesting protocol for research due to the applications that take advantage of its features.

Table 3.2: Main characteristics of the identified protocols.

Protocol	**Architecture**	**QoS**	**Security**	**Resource Requirements**
AMQP	Publish/Subscribe or Request/Response	3 levels	TLS and SASL	High
CoAP	Request/Response	1 level	DTLS	Low
MQTT	Publish/Subscribe	3 levels	TLS	Low
XMPP	Peer-to-Peer or Publish/Subscribe	Not included by default	TLS and SASL	High

Based on this discussion, MQTT will be addressed in greater detail in the following section.

3.4 Message Queue Telemetry Transport protocol (MQTT)

The MQTT [61, 79] is a messaging protocol that works at the application layer, developed by Andy Stanford-Clark (IBM) and Arlen Nipper (Arcom, now Cirrus Link) in 1999. It is currently widely used in the IoT industry due to its low computational requirements, due to this industry being mostly composed of small devices with limited resources.

3.4.1 MQTT architecture

This protocol contains a hierarchy of topics to store received messages. In other words, it is possible to organise the received data according to the type of information, such as the temperature measured by a room sensor ("house/bedroom/temperature"). The message transmission works through a *publish/subscribe* model, where communications are asynchronous. There are two types of message:

- ***Publish* messages**, are used to send data from a device to the destined topic.
- ***Subscribe* messages**, used by devices to *subscribe* to the desired topic/s. If they have successfully subscribed, they receive the data whenever a *publish* message is sent to the subscribed topic/s. Several devices can be subscribed to each topic, allowing communication from 1 to "n" devices (similar to multicast).

  MQTT clients can act as subscribers, publishers, or both.

One of the most important features of MQTT is the QoS, which can be configured according to the application needs. It consists of three levels:

- **QoS0**, messages are sent in a best-effort approach and are sent only once. Therefore, reception is not guaranteed or acknowledged.
- **QoS1**, uses the acknowledgement mechanism to verify that the recipient has received the message by receiving a "PUBACK" message (sent by the recipient). It confirms that the recipient has received the message. If it is not received, the message sender sends the message again until the reception is confirmed. This scenario can increase network congestion. If it is a network with many devices that communicate and low bandwidth, it can also cause the receiver node to receive duplicate messages.
- **QoS2**, makes sure that the message is only received once by the receiver, thus avoiding the duplication of messages in the reception. Although, it increases the overhead in communications. This approach uses the four-step handshake mechanism, as illustrated in Figure 3.1. Initially, the publisher sends the QoS2 type *publish* message. If the receiver receives the message, he replies with a PUBREC (publish message received), which confirms the reception of the message. Otherwise, the publisher resends the publish message with a DUP (duplicate) flag until it receives the PUBREC (response of the PUBREC packet) from the receiver end. Upon receiving the PUBREC, the publisher discards the initial PUBLISH message and stores the receiver's PUBREC message. The publisher then sends a PUBREL message. Receiving this message, it discards all stored states and sends a PUBCOMP (response of the PUBREL packet) message to the publisher. The receiver stores a reference to the packet identifier of the original PUBLISH packet to avoid processing the same message a second time until the receiver device completes this process by sending a PUBCOMP packet back to the sender. Finally, the sender also discards all stored states when it receives the PUBCOMP message.

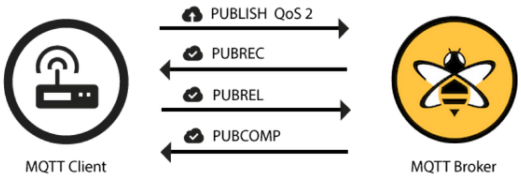

Figure 3.1: Four-step handshake of MQTT's QoS type 2 messages [70].

Other features of the MQTT are:

- **Retained messages**, which consists of keeping the information on the topic even after it has been sent to all subscribers. If there is a new subscription in the topic, the message retained is sent to the subscriber.

- **Wills**, if the device disconnects from the broker, a specific message is published to a specific topic. For example, to specify that the device is currently offline.
- **Bridges**, it allows several brokers to connect. It aims to share information on common topics between brokers that are in different places.
- **Clean session/Durable connections**, in the case of clean sessions, when the device disconnects from the broker, its subscriptions are removed. Contrary to durable connections, its subscriptions are saved, and any QoS 1 or 2 messages are stored until the device reconnects.

The MQTT broker manages all these functionalities mentioned. The broker is responsible for managing the topics, like accepting messages from each topic and their distribution to subscribers.

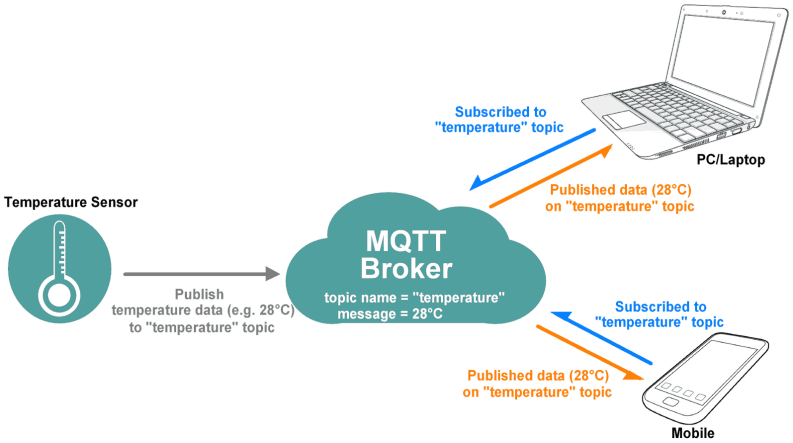

Figure 3.2: Architecture of the MQTT *publish/subscribe* model [92].

In Figure 3.2, it is possible to visualise a typical MQTT architecture. Where sensory devices send information through *publish* messages to a topic in the broker. In this case, it sends information about the temperature of the environment where it is located. The broker stores this information in the target topic: "temperature", and then sends it to the subscribers of that topic, which can be actuating devices (air conditioning) or more complex devices that perform data analysis.

3.4.2 MQTT applications

The applicability of this protocol in IoT applications is due to some of its characteristics:

- *Lightweight*, i.e., it requires little bandwidth to transmit data and few computational resources. It allows resource-constrained devices and networks to transmit data with little delay and low energy consumption, maximising energy consumption efficiency. Which is important for devices that are powered by battery systems.
- Message delivery assurance, through the use of QoS (QoS 1 or 2), it is possible to guarantee the data delivery between an MQTT node and the broker. This feature is important for unreliable networks with packet loss (e.g., long-range networks like LoRa networks).

- Interoperability allows heterogeneous devices to communicate with each other, regardless of whether they are devices with limited resources or servers in the cloud that host applications where the data is being used.
- Scalability, it is possible to expand the number of devices on the network. An important factor in an IoT network where many devices are needed and need expansion.

These MQTT features make it a useful protocol for some types of IoT applications, such as:

- Applications with mobility, such as the automotive industry [69], suffers from problems related to the reliability of the connection of nodes to the network. This problem is due to the mobile network (GSM, UMTS and LTE) locations with no coverage (blind spots). The device's connections to applications in the cloud can be interrupted, which results in packets being lost or not transmitted. With its QoS1 and 2 mechanisms, MQTT allows solving this problem through reliability in data delivery. Additionally, Durable Connections can store the device's subscription/s and retransmit the data that was not received or sent.

  These applications with mobility also require the privacy of their users, especially their location and their data. In a scenario where a vehicle user provides information about an event in his location, if the data is linked to him. and not stored or used privately and securely, it can be used to track and profile the user.

- Applications with a large footprint that use long-range communication channels, e.g., LoRaWAN and SigFox, are networks with very limited bandwidth. However, since MQTT is a protocol that has small packets (in some cases, it can have only 2 bytes), compared to other protocols like HTTP (can have up to 8 kilobytes), the communication delay is lower.

In the survey [85], an analysis of the MQTT protocol is made, and some applications for which MQTT is suitable are identified. Some applications are included in the health area (monitoring patients with chronic diseases outside the clinics through sensors), in the electrical energy area (electricity meters that transmit data read by the network), and social networks (for example, a lightweight approach for transporting messages between people within a company). Finally, some problems of the MQTT are mentioned: lack of a data expiry mechanism (currently implemented in version 5.0 [79]), lack of security mechanisms, absence of message ordering mechanisms and lack of priority functionalities.

In [87], is proposed a health application, where devices that read telemetry data about the health state of patients (e.g., blood pressure, scales and motion sensors) with chronic diseases are placed in their homes. The data is sent to the cloud management application through a middleware composed of an Simple Network Management Protocol (SNMP) agent and a proxy, which obtains the data from the devices and routes it to the target application. In the management application, the data is processed, and if there is any problem with the patient, an alarm is generated so that healthcare professionals can react accordingly. To ensure some level of privacy, they use access and authentication policies for professionals to access patient data. For example, the healthcare professional can only access the patient's data when he/she is at the patient's home and has an appointment.

In the article [86], a Smart Parking application is proposed for an airport. The user can access the location of his vehicle in the car park through a web application that shows the GPS location on a map. The administrator can only view the information of the user and

his vehicle. However, there is not any proposed security mechanism. This approach does not ensure the privacy of the user's data due to being dependent on authentication. I.e., if it is poorly implemented (insecure credentials) and without proper security mechanisms (TLS connections between the administrator or client and the web application), it becomes vulnerable.

In this Smart Parking proposal [68], the authors use MQTT by implementing the Mosquitto broker in order to transmit information related to car park occupation. An ultrasound sensor is used to verify if the slot is occupied or not and transmit the information to a web application, where a car park map with occupied and free spaces is shown. They argue that when drivers look at the car park map and the location of free spaces, they emit less CO_2 because they can outline a route, so they park faster and avoid driving extra kilometres to park the vehicle.

3.4.3 Security in MQTT

The MQTT has some mechanisms that offer data security and authentication. It offers identity through IDs for each MQTT client, which allows controlling access to topics. However, if an attacker uses the ID of a legitimate MQTT client, he can access the topic and prevent that same client from subscribing to that topic, causing an attack on the client. Therefore, user authentication is provided by the protocol. However, the transport of the credentials is made in clear text format if no encryption mechanism is implemented in the transport. It supports TLS, which allows encrypting the communications between the broker and MQTT clients through symmetric encryption (uses a secret key shared between the sender and receiver in establishing the connection). This protocol offers confidentiality and data integrity. Although, they require a high level of resources, which is not feasible for devices with limited resources, and not all of these devices support it.

The MQTT lacks security and privacy mechanisms, which is unsuitable for applications with resource-constrained devices where the confidentiality, privacy, and integrity of sensitive data are fundamental. For example, applications where the privacy of users, location and data is essential to prevent tracking and profiling attacks.

The article [53] focuses more on MQTT security problems, demonstrating some possible attacks in the communications between the broker and the MQTT client. These attacks take advantage of vulnerabilities in confidentiality, integrity, availability, authentication and authorisation. They mention some proposals which implement security mechanisms to mitigate some of the addressed MQTT problems. Finally, they propose some issues that should be taken into account to make IoT architectures more secure, such as privacy and data integrity, so that users' sensitive data is secure and unchanged.

3.5 Blockchain

Blockchain [82] is a distributed database that has a continuously growing list of records organised into entities called blocks. The blocks are "chained" together by connecting each block to the previous block's hash. The concept of blockchain was first studied in 1991 by Stuart Haber and W. Scott Stornetta. Although, an anonymous person defined the first blockchain proposal by the pseudonym of Satoshi Nakamoto, who published a paper in 2008 [77], talking about bitcoin, which later became an actual public blockchain. This technology has much potential in various applications, especially in the IoT industry,

mainly because of its decentralised, "trustless", privacy and security features, which could improve applications' reliability.

3.5.1 Blockchain architecture

Blockchain provides a decentralised, distributed, immutable, auditable, transparent, and secure ledger. Its decentralised and distributed architecture does not use a single authority (intermediary) to verify transactions, eliminating single points of failure and third-party dependencies. The nodes of the network communicate with each other in a peer-to-peer model. A consensus (agreement on the current state of the blockchain) is used to verify the transactions made, so the network's trust is distributed among the participating nodes. The majority of participants validate the transactions, which removes the need for a central authority to validate them and allows to automate the process.

Data stored in the blockchain is considered tamper-evident and resistant because it is computationally hard to change the content of a block. An attacker, to be able to tamper the block's content (which alters the structure of the blockchain), needs to control 51% of the resources (nodes) in the network [82], to agree and validate the changes made to the state of the blockchain. It is considered computationally and monetarily expensive and unfeasible to achieve.

Nowadays, most blockchains support smart contracts [82] that can be used to validate transactions based on predefined conditions. This technology allows the integration of decentralised applications into blockchains through functions and defined conditions. They can be used to provide authentication and authorisation for certain transactions, store information and trade assets or data between two participants in an automated and secure manner. Both participants have to comply with the predefined clauses (similar to a real contract) to be successful, e.g., transferring a certain amount of cryptocurrency as an assurance to trade the assets. If something is wrong with the assets (invalid data or incorrect amount of cryptocurrency), the contract request is revoked.

There exist various types of blockchains: permissionless (i.e., public blockchains) and permissioned (i.e., private blockchains) blockchains. Each one of them has its use cases and functionalities (detailed further ahead). For example, permissionless blockchains can be consulted and accessed openly and fully, which provides transparency. In contrast, permissioned blockchains are closed to the public, only a set of nodes can participate, and restricting policies can be imposed inside it.

Composition

The blockchain comprises various blocks linked together, which store information about them, transactions and other important data. Such blocks contain a unique hash produced by hashing all its content (e.g., transactions and timestamp of creation), also known as a digital fingerprint. This hash is used to chain the blocks together, as illustrated in Figure 3.3, the following blocks in the chain have a field parameter that identifies the hash of the previous blocks. Suppose that the content of a block is modified, its hash changes, which results in a broken chain. Thus, the state change is invalidated by the majority of the participants of the blockchain. This block "chaining" by the hashes of the blocks is what provides immutability to the blockchain.

Since the first block of the chain (genesis block) does not have a previous block to point to, the previous hash value is hardcoded (e.g. zero).

The transactions are typically stored as a Merkle Tree structure [59]. This structure helps to verify the integrity of the data content by hashing the nodes with the hash of the child node. In Figure 3.3, the transactions are hashed in a leaf node, which, consequently, are paired together in a parent node. All hash transactions are combined in a single hash stored in the Merkle Root, present in each block. It allows the nodes of the blockchain to validate individual transactions without having to validate the entire block.

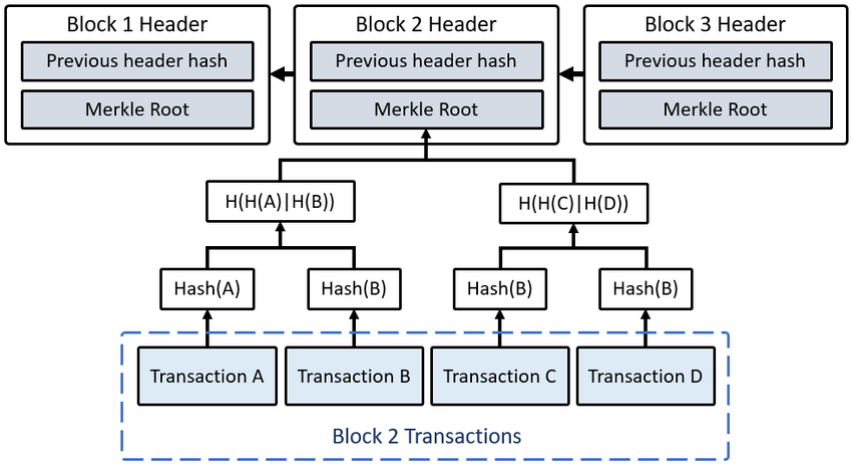

Figure 3.3: Architecture of blocks in the Blockchain [59].

Blockchain addressing system

In order to perform transactions, the blockchain participants need a blockchain address and a key pair (public and private key). The private key (a unique 32-byte alphanumeric string) is generated automatically by a blockchain wallet, which stores the users' key pair and manages their cryptocurrency. The private key is used to create the public key and sign transactions (digital signature) to provide authenticity of the transaction sender. The cryptographic algorithm used to generate public keys depends on the blockchain used, e.g., Bitcoin and Ethereum use Elliptic Curve Digital Signature Algorithm (ECDSA). ECDSA is commonly used, because it's faster and computationally less demanding than Rivest-Shamir-Adleman (RSA) [6]. Multiple public keys can be derived from the private key. However, the private key cannot be derived from the public keys due to the one-way cryptography used.

With the keys generated, it is necessary to create a blockchain address to perform transactions. The blockchain address is obtained from the public key using a one-way cryptographic hash function, which is different in each blockchain. In Bitcoin, the algorithms used to generate an address are the Secure Hash Algorithm 256 (SHA-256) and the RACE Integrity Primitives Evaluation Message Digest 160 (RIPEMD-160). The format of the address varies from blockchain to blockchain. In Bitcoin, the address is 25 bytes long with a Base58 format. Ethereum addresses are 40 hexadecimal characters (20 bytes), prefixed by an "0x" to identify that it is in hexadecimal format.

When the user has all keys and blockchain address created, the only requirement is to have cryptocurrency of the specific blockchain to pay fees for the transaction creation. Moreover, certain wallets can have multiple blockchain addresses, each one corresponding to a different blockchain.

Blockchain types

Two main blockchain types exist. Each one of them has different features, benefits, and use cases. The types are:

- **Permissionless Blockchains**, also known as public blockchains. They are open, and do not have any access control mechanism, since any node can participate in them. The nodes can read, write, and participate in the consensus without requiring permission or trust from other nodes in the network. Additionally, all transactions are publicly accessible. The consensus used is more complex than in the other types of blockchains to maintain the blockchain secure. The most used ones are Proof of Work (PoW), Proof of Stake (PoS) and Proof of Burn (PoB) (detailed later in this section).
- **Permissioned Blockchains**, are private blockchains governed by a single or multiple entities, where nodes have to be authorised to be able to participate in it. Each node knows and trusts every other node, and any new node added to the network needs to be allowed by the majority of nodes in the network. The nodes have different permission levels. Since this blockchain uses a trust model the consensus can be simpler and personalised to fit the purpose of the blockchain. They provide granular control of the nodes, are faster, more efficient, and more cost-effective than public blockchains because of the reduced number of nodes and reduced chain size. Fees are not imposed in creating transactions since it is not computationally expensive to create new blocks, and its cryptocurrency does not have market value.

  These blockchains could be used in the private sector where the enterprise controls the resources, needs to know the identity of the nodes and control the access, such as the military and government applications.

Types of nodes

Blockchain is composed of different types of nodes responsible for storing the chain blocks (decentralised and distributed), creating transactions, adding new blocks, and validating them. The nodes connected to the blockchain can be any device, e.g., a dedicated server, personal computer, or a device with fewer resources (e.g., smartphone or Raspberry Pi). As long as it supports the technology. The nodes ensure that the blockchain's data is valid, secure, and accessible to authorised parties. Moreover, the usual types of nodes are:

- **Full node**, it stores all blocks of the chain, which are verified, authenticated, and stored by the full nodes in a network. When a new block is accepted into the blockchain, the full nodes save and store it on top of the existing blocks. They can also send new transactions and hold wallet balance information. This type of node requires high storage capacity and network bandwidth because the blockchain can reach enormous sizes after some months or years of usage. Bitcoin is currently (10th of November 2020) at 309 GB in size and Ethereum at 179 GB [28].
- **Light node**, this type of nodes are similar to full nodes. However, they do not store the entire blockchains, only the block header. Nevertheless, they can validate transactions in the blocks as well. Light nodes require much fewer resources, which are ideal for constrained devices.
- **Mining node**, are responsible for creating blocks. They create blocks by grouping valid transactions and by solving a computationally hard problem that allows

consensus throughout the network (e.g., PoW in bitcoin). The creation of blocks is called mining because the cryptocurrency rewards given to these nodes in this process resemble discovering/mining raw materials. Mining nodes can work individually or in a pool of nodes. These nodes require high computational resources and must be energy efficient to be profitable.

Smart contract

The concept of smart contract was defined by Nick Szabo in 1993, who characterised it as: "A computerised transaction protocol that executes the terms of a contract" [82]. Some blockchains, such as Ethereum, integrated this concept for transaction assurance based on predefined assets, such as exchanging information. The smart contract assures that both participants receive the correct amount of cryptocurrency or data from each other. Essentially, it is a set of executable code that runs on top of the blockchain that facilitates, executes, and enforces an agreement between untrusted parties, without the need of a trusted third party [89]. The smart contract code is stored on the blockchain, and a unique address identifies each contract. To operate with it, nodes of the blockchain have to make a function call addressed to the smart contract. Depending on the function, the node could pay a fee to the smart contract to be successfully executed.

When deployed in the blockchain, the smart contract cannot be modified by any entity, even by its developer. As it is stored in the blockchain, its content is immutable. The smart contract can provide enhanced Security and trust [36], since it remains immutable on the blockchain. Thus, it is not possible to change the code inside a smart contract. Furthermore, using them to exchange data or currency can prevent errors or malicious actions, e.g., human errors or corruption among institutions.

Consensus protocol

For blockchains to work in a decentralised and self-regulating model, they need to define a trust mechanism. The trust mechanism used is called consensus protocol, which is how the participants in a distributed system agree on a common state of the system. The consensus mechanism allows the participants to maintain an identical copy of the blockchain. Otherwise, they would end up with conflicting information, undermining the entire purpose of the distributed network.

Private blockchains require specific consensus protocols, which can be simpler since every node is trusted and known. The Practical Byzantine Fault Tolerance (PBFT) consensus algorithm is an example of a consensus mechanism used in some private blockchains. It reaches a sufficient consensus despite malicious nodes in the network failing or sending out incorrect information. The consensus consists of the majority of nodes agreeing on the state of the blockchain, which is based on state machine replication and replicates voting for consensus on state change. Proof of Authority (PoA) [15, 93] works with a trustful model, which is based on reputation. The validator nodes do not stake coins. They stake reputation. Other validator nodes arbitrarily select validator nodes. Thus, not every participant can validate blocks. It is more oriented to private blockchains where a limited number of validators are present. It is currently used in Microsoft Azure Blockchain Service [1] and can be used to create a private blockchain with [14]. It offers high throughput and scalability [15]. Although validators' identity is publicly accessible, which threatens their privacy and security, and is less decentralised.

However, public blockchains require more complex consensus protocols, as anyone can join the network anonymously. Some consensus protocols used in public blockchain are [67]:

- **Proof of Work (PoW) [24, 82, 93]**, uses computational resources from miner nodes. The miners have to solve a computationally demanding problem that is hard to solve but very easy to verify to create a block. When a miner node solves the problem, the new block is spread across the network to be verified and appended to the blockchain by other nodes. The miners are rewarded in cryptocurrency of the blockchain at hand. This consensus is used in well-known blockchains, such as Bitcoin [77] and Ethereum [57]. In Bitcoin, the frequency of block creation with PoW is 10 minutes. However, this consensus protocol is less scalable (low throughput), requires more computational resources and is more prone to 51% attacks [82].
- **Proof of Stake (PoS) [16, 82, 93]**, is based on the participants who own more coins are more suitable to add blocks to the blockchain since they are more interested in the survival and the correct functioning of the system. The consensus algorithm chooses randomly the participants nodes that add the blocks to the network based on the amount of stake they hold in the network. When a node gets chosen, it checks if the transactions in the block are valid, signs the block and adds it to the blockchain. As a reward, the node receives the transaction fees associated with the transactions in the block. In this consensus, there is no concept of miner nodes. It does not take into consideration external resources but, instead, internal resources (cryptocurrency). It is currently used in a vast number of blockchains, such as Cardano [38] and Ethereum 2.0 [48] (Ethereum upgrade that will integrate PoS to enhance its performance and reduce the cost of its transactions).

  This protocol is more energy and resource-efficient than PoW since there is no need for high-powered mining farms. Furthermore, it is less prone to 51% attacks due to the economic infeasibility of buying 51% of the current circulating supply of a cryptocurrency [16]. A common critique of this protocol is that it promotes enrichment of the rich [82], and it requires incentives for the nodes to vote on the correct block (e.g., slashing [19]).

The article [82] explores the concept of integrating Blockchain into IoT applications. This integration aims to provide a distributed, immutable, transparent, secure, and auditable ledger. It aims to solve problems related to the privacy, security, and reliability of IoT applications. The Ethereum blockchain was identified as one of the most prominent blockchains to be implemented in IoT, mostly because of Smart Contracts that enable implementing policies and measures. However, not every Blockchain is suitable to be used in a constrained network due to some challenges related to legal issues, resource overhead, and being attached to currency. The authors tested some blockchain platforms that are compatible with the Raspberry Pi [21]. The results showed that they do not significantly increase the overhead in terms of resources (except for the full node versions that are not suitable for constrained devices). The authors conclude that this technology will revolutionise IoT, although some challenges identified in this article must be considered before implementing the two together.

3.5.2 Blockchain in IoT

The majority of IoT applications use a centralised architecture. However, this approach leads to vulnerability, such as single points of failures, lack of reliability and resilience.

Blockchains can solve these problems and enhance the reliability and resilience of IoT applications. Furthermore, it enhances the privacy of the users. The main benefits of this integration include [82]:

- **Decentralisation and scalability**, since blockchain uses a decentralised and distributed peer-to-peer architecture, there is no need for a central authority to validate the transactions and audit the system, which eliminates single points of failure. Additionally, it could enhance the scalability of IoT applications since it is distributed. However, blockchains require higher throughputs to accommodate the number of events generated by these applications.
- **Authentication and authorisation**, mechanisms of authentication and authorisation can be implemented with tamper resilient properties using smart contracts. They can be used for authorisation to define who is allowed to access certain information stored in them (specific service providers or other users). In addition, permissioned blockchains can provide more granular access control rules on the participants of the blockchain.
- **Privacy**, blockchains provide anonymity to the user, either by using a unique blockchain address each time it makes a transaction [63] or by using a blockchain address that is not associated with any personal information of the user [56]. However, this depends on the implementation of the blockchain, e.g., some blockchains, such as Stellar [13], implement Know Your Costumer (KYC) or Anti-Money Laundering (AML) approaches for security reasons, which diminishes the user's privacy.
- **Security**, information remains immutable on the blockchain, and the nodes can check its integrity. Although, this feature can be a drawback to applications that handle user sensitive data since it does not allow data erasure and modification, which violates GDPR. Furthermore, each node uses his private key to sign his transactions, providing authenticity to the transactions.
- **Resilience against Denial-of-Service (DoS) attacks**, since transactions require a fee in cryptocurrency to be created, it is monetarily infeasible for the attacker to congest a blockchain.

As mentioned before in 3.5.1, some blockchains can create decentralised applications through smart contracts, which is useful for decentralised IoT applications. Some of them are: Ethereum [57], and HyperLedger Fabric [52].

Ethereum can be used as a public or private blockchain, which provides flexibility for applications. His consensus protocol is PoW (in the current version, before Ethereum 2.0), which supports smart contracts, enabling blockchains to move away from only being used for cryptocurrency exchanges. It is one of the most popular platforms to develop applications [82]. An example of an application is: ADEPT [90], developed by IBM, which aims to enhance Machine to Machine (M2M) communications and device autonomy with smart contracts. A possible use case, identified in [90]: allow washing machines to buy automatically detergent from retailers when needed.

HyperLedger Fabric is a private and modular blockchain designed to be used by companies. It provides identity control service and access control lists through private channels to control and restrict access to their shared information in the network. It is used in IBM's blockchain platform [32], which could be used in food supply chains (Food trust [33]) to improve traceability, with the intent to improve the safety and quality of food and efficiency in the supply chain.

Blockchains can be applied in IoT applications in the Automotive Industry [58, 66]. It can protect sensitive data, such as vehicle owners information, location information, and *biotelemetry* (e.g., heart rate, temperature, blood pressure) from sensors. Permissionless blockchains can be used in Social Internet of Vehicles (SIoV) applications, where smart cars are connected to share data (e.g., accidents, traffic data, or best routes) and communicate with service providers (e.g. insurance companies). The users in these kinds of applications can remain anonymous. However, companies could use private blockchains, where participants are known, for fleet management (monitor vehicle telemetry for maintenance and consumption purposes), monitor the driver's health and data traceability. Renting or sharing smart car companies could benefit by building a sharing economy where each car can be rented securely and quickly without the need for any authority. Moreover, KYC mechanisms could be implemented to monitor and classify the clients' behaviour while driving.

Another suitable blockchain methodology for IoT applications that generate sensitive data and large data is the off-chain blockchain. This type of blockchain, as the name suggests, stores data outside the blockchain. Inside the blocks only resides pointers or anchors of the data to verify the integrity and time stamps of the data [49]. It could improve the performance, scalability and privacy of data in applications that work with highly sensitive data in large amounts (e.g., healthcare) by ensuring that only authorised entities can access it.

In [66], the authors address the feasibility of using blockchain in the automotive industry. Challenges related to the financial sector, privacy and security in the automotive industry are identified. The authors provide some guidance to assess whether blockchain is a suitable solution for specific applications. A list of blockchain-based applications is provided, where blockchain can exchange trusted and cyber-resilient information in the automotive industry. However, some blockchain weaknesses are identified related to its complexity, lack of maturity in the industry, interoperability, lack of standardisation, regulatory and legal aspects, governance and cultural acceptance.

The survey [58] focus on exploring privacy issues and possible solutions in the SIoV using blockchain. SIoV has the intention to establish a social relationship between vehicles and service providers to share telemetry data, traffic data, user data, or other types of useful data. Despite the increased data availability, connectivity, and autonomy of vehicles in SIoV pose a major threat to the privacy of its users. The authors identify seven dimensions of privacy in the SIoV environment and the privacy threats in each layer of the application (physical world layer, gateway layer, fog layer, and cloud layer). Finally, the authors identify and discuss some SIoV blockchain-related privacy proposals.

3.5.3 Blockchain limitations and vulnerabilities

Despite the strengths and benefits of blockchain applications, it has some weaknesses that need to be taken into consideration, which are:

- **Does not make it easy to implement GDPR required mechanisms**, due to the immutable nature of blockchains, its users are not allowed to manage their data or information present in it, such as modifying or deleting. This impairment violates the GDPR, which, by default, is not a viable technology to work with users' sensitive data. Another parallel solution to store data is required to store or process this kind of information to mitigate this limitation.
- **Scalability and size growth**, the increase of the blockchain size and network size

results in the degradation of its scalability and increases its storage capacity requirements. New nodes require more time and storage resources to fully synchronise with the current blockchain's state [93]. Additionally, as the usage increases, the congestion also increases. Permissioned blockchains attenuate this problem by having a smaller network size and simpler consensus protocol. A possible solution for permissionless blockchains could be the use of an off-chain approach to improving the scalability.

- **Smart contract flaws** [75], since smart contracts are composed of code, and humans code it, it is prone to flaws. These flaws can be exploited by malicious actors to their advantage, either to obtain sensitive information or theft. As it is standard in software development procedures, smart contracts should be audited, tested, and formally verified. Using well-established libraries can also help mitigate this risk. Furthermore, data integrity at the smart contract level may be achieved by establishing permissions to prevent unauthorised participants from accessing and modifying information.

- **Vulnerabilities**, blockchains are vulnerable to certain attacks. Some of them are:

  - The **51% attack** [82] consists on the attacker having 51% of the mining power (in PoW), or 51% of the stakes (in PoS). This attack can compromise the integrity and reliability of the blockchain. However, this attack is computationally and financially hard to achieve, but not impossible. The blockchain should be more decentralised and have many trustworthy nodes to maintain the blockchain secure against this attack. Smaller permissionless blockchains can be more vulnerable to this attack since fewer resources are needed to accomplish it.

    If an attacker performs this attack successfully, he can change the structure of the blockchain by mining the blocks that he desires and publishing them, forking the chain, thus discarding all previous transactions made. With this amount of power, the attacker can select the blocks to be accepted into the blockchain, putting at stake the security of the blockchain by disrupting its tamper resilient properties.

    Permissionless can mitigate this attack since non-cooperating nodes can be removed from the pool of block validation nodes or the network.

  - The **double-spend attack** [82] is an attack where the same cryptocurrency is used twice. This attack is possible based on the fact that cryptocurrency is digital. Thus, it can be copied and broadcasted multiple times. However, the attacker needs to have 51% of the network power to perform this attack to validate invalid transactions where cryptocurrency is used multiple times. If this attack is successful, it will affect the reliability of the blockchain.

  - **Sybil attack** [54] is another vulnerability where a single entity controls multiple fake identities. It can influence the blockchain network by having more power, influencing the information reaching other network nodes. This attack can be mitigated by increasing the difficulty of accessing the blockchain network (invite only) or creating new accounts (one computer per account) and having weighted power based on reputation.

  - **Linking attack** [95], is an attack where the attacker tries to identify the real ID of an anonymous blockchain user by establishing a link between multiple transactions with the same blockchain address. This attack violates the user's privacy. The attacker can trace data generated by the user and gather information about him (user profiling). This attack can be mitigated by renewing the blockchain address of the nodes frequently, which is associated with the

transaction made by them. Alternatively, hiding the address of the actual user could also safeguard its identity. Zero-knowledge proofs could also be used to obfuscate the details of blockchain transactions [75].

- **Private key compromise** [75], blockchain users own a private key that works as an identifier. Therefore, owning this private key is the same as owning the user's identity, which imposes a major threat to the users' security. Thus, it is essential to keep these keys secure. They can be compromised due to errors in key generation, storage, human errors, or stolen by malicious actors. Therefore, these keys should be used and stored securely. However, recovery mechanisms should be implemented to enable a user to regain control of an identifier in loss or theft cases.
- **Quantum computation** [75], computers with extreme amounts of computation (quantum computers) are currently being developed, which are foreseen to disrupt the security of modern cryptography algorithms. Furthermore, it represents a threat to the security of blockchains since it is dependent on asymmetric cryptography. Thus, the security of the users' private keys could be at stake. A possible solution for this future hypothetical problem is developing and standardising quantum-resistant algorithms that may help alleviate the threats borne by quantum computers.

3.6 Conclusions

This chapter introduced privacy in IoT. It can be divided into four areas: user privacy, location privacy, data privacy and user-controlled privacy. User privacy consists of the privacy of users' sensitive data (e.g., biotelemetry data). If its privacy is violated, it can endanger him by allowing external entities to perform user profiling and user tracking. Privacy of location, such as geospatial data from smart devices (e.g., smart cars), is important to prevent tracking. Data privacy consists of the privacy of all remaining data generated by IoT devices to prevent attacks that could gather sensitive information by linking the data entries. Finally, user-controlled privacy allows the user to define privacy preferences over his data and offers him more control over his data.

Various transmission protocols were addressed with some level of detail. However, MQTT emphasised as the most appropriate for IoT applications that use location data, which is more appropriate to the area of interest of this thesis. MQTT provides lightweight messaging, resilience, and QoS features to improve the performance and resilience of constrained IoT communications. However, it lacks adequate privacy and security features for constrained devices since its confidentiality mechanisms require some computational resources to be effective.

Blockchain was identified as a possible solution to provide enhanced privacy and security to IoT applications. Based on its decentralised and distributed architecture, data immutability, the anonymity of users, and resiliency to single points of failure. Smart contracts are one of its most interesting features since it allows creating decentralised applications, which can be used based on the needs of the IoT application. As mentioned before, one possible use is to offer a decentralised authorisation and authentication mechanism for access control.

In the next chapter, the concept of privacy in IoT will be addressed to instigate the need to provide privacy in IoT applications. Privacy-enhancing proposals related to MQTT and blockchain will be described, as well as their privacy and security mechanisms.

Chapter 4

State-of-the-art

In this chapter, the main subject of the thesis will be addressed: privacy and location in IoT. Privacy proposals related to MQTT and Blockchain from the literature will be identified and classified based on their privacy and security mechanisms. This chapter also aims to identify the mechanisms and solution to be used in our proposal.

4.1 Privacy solutions

In the survey [81], the authors address privacy in IoT. Some applications are mentioned where privacy is an important factor and the possible risks if privacy is compromised, e.g., in eHealthcare, Smart Homes, Public Safety, and Stock Management. They discuss four key aspects of privacy (user privacy, data mining, underlying technologies, and legal standards), pointing out their problems and challenges. Based on the identified privacy issues, the authors propose a privacy framework for IoT applications to be used as a guideline to classify the provided privacy. Some existing privacy solutions (Privacy-Enhancing Technology (PET)s) are identified, addressing their strengths and weaknesses. Finally, they conclude that the Privacy by Design (PbD) is a potential approach to privacy in IoT. However, it is still underdeveloped and needs research and solutions in some aspects. Which are: defining a general model for privacy, developing PETs based on PbD to allow scalability and heterogeneity, integrating solutions with a perfect balance between policies, location requirements and access control mechanisms for sensitive data.

As mentioned in [81], the authors propose a privacy framework for IoT where they identify important technical characteristics, which are:

- **Openness, transparency, and a specified purpose**, clients should be aware of the information collected in the applications, the purpose of collecting that information, other parties who will have access to the information, and how that information will be stored.
- **Identity privacy**, it should not be possible to profile or track users based on their identities.
- **Temporal and location privacy**, it should not be possible to track or profile consumers based on events or geolocation.
- **Query privacy**, it should not be possible to profile or identify users based on the queries they make to service providers.

- **Access control**, users should have control over the access control policies. They should be able to tune the granularity of data access depending on the users and queries.
- **Interoperability**, enabling cross-border support of privacy policies among different technologies, standards, and legislation.
- **Data minimisation**, collect data in lawful and fairways and limit personal data collection to data needed to perform a given service.
- **Accountability**, the consumer and service provider should agree about the controllability and visibility of the service provider's responsibility concerning the given service or information.
- **Security of data**, provide security to sensitive data by preventing loss, unauthorised access, data tampering, or disclosure.

In the literature, there exist various PETs in the area of IoT. However, MQTT and blockchain will be the main subjects of this work, with the intent to propose a privacy-enhancing solution. PETs present in the literature cover the areas of privacy mentioned in 3.1, using distinct privacy mechanisms:

- **Access control** ([49, 62, 63, 76, 78, 95], which provides privacy by controlling the access to the data stored. Despite this, if the data payload is not encrypted during transmission, an attacker can eavesdrop it. This mechanism provides some level of privacy in terms of data privacy and user privacy.
- **User anonymity** [63, 76, 95], most PETs that use blockchain use this mechanism since the user's ID is hidden with the use of public addresses obtained from its key pair. Thus, this mechanism prevents user profiling and user tracking.
- **Topic obfuscation** [65], with this mechanism, the topic names of the MQTT system are generated with a hashing function, which prevents topics analysis by their names. As a result, the type of data published on the topics is very hard to be traced. For this solution to be even more effective, the data must be encrypted. Nevertheless, this mechanism can improve data privacy since the attacker does not know what kind of data it is.
- **Lightweight cryptography** [62, 84], this approach provides confidentiality of the data by using a lightweight cryptographic algorithm to be used in communications between constrained devices. Some privacy is added to the data.
- **Blockchain** [49, 56, 62, 63, 76, 95], as mentioned in chapter 3.5.2, this technology can be used to provide privacy of user's identity and data. Smart contracts are also used to improve the security of the data by providing access control.

The PbD is a methodology that has been increasingly addressed. It consists of a methodological engineering requirement in which privacy requirements are identified and considered as objectives of a project [81]. It consists of anticipating and preventing attacks that compromise data privacy in the future, incorporating privacy into the project design. This methodology is not a concrete privacy solution but can be used as a framework for application development.

4.1.1 General privacy proposals

MQTT has an important role in our proposal since it is useful for IoT applications with mobility. Ergo, this section presents privacy proposals that use existing mechanisms to provide privacy of data implemented on MQTT communications. In Table 4.1, a summary of the proposals and some of their characteristics are identified.

The article [78] proposes a security mechanism to protect the data generated by IoT devices in order to preserve the users' privacy. The authors have developed a solution using a security toolkit called SecKit to enforce security policies. The solution has been integrated into an MQTT broker, Eclipse Mosquitto, through a security plugin. This solution allows controlling the access of MQTT message types: *connect, publish and subscribe*. Depending on the condition defined in the Access Control List (ACL), an action can be allowed, denied, modified, or delayed. Through experiments, the authors concluded that the solution developed is feasible for environments with limited resources due to adding, on average, a delay of only ten Milliseconds (ms).

In [84], it is proposed an adapted Attribute Based Encryption (ABE) based on Elliptic-curve cryptography (ECC) integrated into the MQTT protocol to enhance its privacy and security. This solution aims to improve the MQTT by providing lightweight cryptography (ECC) and access control (ABE), which is ideal for IoT applications. The authors tested the solution by comparing it to a similar one that uses Advanced Encryption Standard (AES) cryptography instead. The result has shown that ECC is faster than AES at encrypting and decrypting data, making it more suitable for constrained systems.

In the article [65], some solutions are proposed to improve data privacy in MQTT communications, with emphasis on hiding the topics to which messages are addressed and encrypting the payload. They performed some performance tests with one solution, namely, Advanced One-Time Password (AOTP). This solution uses a technique to obfuscate the topic names using a hashing function that creates a hash table for each topic. It defines who can subscribe to the topic, in which they compute the hash with the hashing function, thus being able to get the message sent by the sender. After each *publish*, the hash is removed from the table, and the hashes are used only once. This solution allows offering data privacy when public brokers are used. Tests have shown that this technique adds little overhead to communications, adding an average delay of 10 to 15 ms.

Table 4.1: General privacy proposals identified.

Proposal	**Description**	**Framework characteristics**	**Privacy area**
Enforcement of security policy rules for the Internet of Things [78]	It's integrated SecKit (security toolkit) with an MQTT broker (Mosquitto) to enforce access control in *connect, publish and subscribe* messages.	Access control; Interoperability; Security of data.	Data privacy.
Secure mqtt for internet of things (IoT) [84]	This solutions aims to improve privacy and security of MQTT, by providing a lightweight cryptography with ECC and access control with ABE.	Access control; Interoperability; Security of data.	Data privacy.
Towards improving the Privacy in the MQTT Protocol [65]	Implements a One-Time Password mechanism to obfuscate MQTT topics with a hash function, only specific subscribers can access the topics.	Access control; Identity privacy; Security of data.	Data privacy.

4.1.2 Privacy proposals with Blockchain

In this section, there are presented privacy proposals that use blockchain technology to enhance the privacy of IoT applications.

The authors of [56] propose a privacy solution that uses One-Time Password (OTP) in the MQTT Broker for authentication and authorisation of devices. The authentication and authorisation of the devices are done in conjunction with the Ethereum Blockchain. It is used as a parallel independent channel to improve the security of this solution. The blockchain is used because it provides a distributed and tamper resilient database and enhances users' privacy. Smart contracts are used to assure the correct functioning of the authentication and authorisation of in-chain mechanisms. The Ethereum users register the MQTT client devices. The users' privacy is preserved since the blockchain does not use the private information of the user. In addition, the devices' IPv6 addresses are obfuscated using a secure hashing function (SHA-256).

In the privacy proposal [63], is proposed a solution to improve user privacy and security in automotive applications using a blockchain-based architecture. The authors have developed an optimised blockchain for IoT applications, called Lightweight Scalable Blockchain (LSB), and proposed an architecture based on it. The LSB allows blocks to be processed faster compared to a bitcoin blockchain. Nodes in the network are clustered (cluster heads (CHs)), which are responsible for managing and performing its core functions on the blockchain, which constitute the Overlay Block Managers (OBM). Transactions are broadcasted to and verified by the OBMs, thus eliminating the need for a third party. This proposal provides authentication and authorisation of the nodes with their pair of asymmetric keys. It provides data integrity with a hash, enabling them to check if the data has been tampered. One interesting privacy feature of this solution is that it enables users to decide the data they want to share with service providers. Accordingly, the users are aware of the data being transferred. Furthermore, the authors identify some applications where their proposed architecture could be implemented, such as remote software updates, insurance, electric vehicles and car-sharing services. In conclusion, this proposal addresses a broad set of privacy and security aspects, making it a comprehensive proposal.

In [76], the authors propose a privacy solution based on an Ethereum blockchain for IoT applications. The solution consists of using a publish/subscriber architecture in a blockchain to be used as a broker. It enables the publishers to control data access to the data sent to the blockchain, which adds a layer of data privacy. The data is encrypted using the public key encryption with equality test (PKEwET), and since data on blockchains remains immutable, the data is protected against tampering attacks. The user's identity also remains immutable on the Ethereum blockchain. The authors did some performance tests that showed that the solution is efficient at encrypting and decrypting data.

The authors in [95], propose a location privacy-preserving solution based on blockchain. Its purpose is to hide the users' location (workers). The workers complete tasks (provided by the requester/s), in their desired area, by going to the locations defined in the tasks. In exchange, they receive cryptocurrency as rewards. This proposal uses both private and public blockchains that trusted participants, named agents, connect. The private blockchains are used to store in a distributed manner the transaction history of the workers across multiple private blockchains, which grants enhanced privacy of the data by preventing reidentification attacks. Considering that only authorised participants can access the private blockchains, the privacy is further improved. The worker's ID is anonymous, and the users' precise location is obfuscated by using an area of X radius. Smarts contracts are also used to assure that the transactions between requester and worker are valid, as-

suring that cryptocurrency is sent and received to validate the location of data provided by the worker and to verify if the tasks are still valid. Compared to typical centralised crowdsensing systems, this proposal proves to be resilient to reidentification attacks since the transactions are distributed across multiple private blockchains, and the users are anonymous. However, the latency introduced by the verification process of creating new transactions could be a problem.

The proposal of [62] consists of using blockchain to offer security and privacy for smart home applications. A public blockchain is used to connect various smart homes, known as Cluster Heads, to allow smart devices from each home to communicate with each other or with service providers that want to access the data. Inside the smart home, it is used a private blockchain that stores the transactions made. Moreover, private and public blockchain blocks contain a policy header to provide access control, providing more privacy to the data. Lightweight symmetric encryption is used in communications to ensure the security of data during transport. The authors identify two attacks that this proposal is immune to: DDoS attacks and linking attacks. As the smart devices are not directly accessible, an attacker cannot install malware on the devices. Even if he manages to infect devices, the outgoing traffic is examined by the policy header, so using devices for DDoS attacks is unlikely. Each device's data is shared and stored by a unique key to protect the system against linking attacks. Finally, the authors examined the solution's performance by doing some experiments and identified that the solution introduces a small delay in the transactions. However, the security and privacy benefits are more relevant.

The concept of tiered blockchain architecture is also proposed in [49]. The system is composed of a consortium blockchain network with multiple private blockchains (identified as private "side-chains"). Both the consortium blockchain and the private "side-chains" have a smart contract with a list of public addresses of authorised devices that can access the IoT data. Contrary to the previous proposals, data is stored in an external decentralised storage system, named Interplanetary File System (IPFS). The private "side-chains" store encrypted IPFS file hashes of the IoT data, which authorised requesters of the consortium blockchain can decrypt. To access the data, the requester sends an HTTP request with the decrypted file hash to the IPFS distributed network, which responds with the data requested.

Table 4.2: Classification of the blockchain related privacy proposals identified.

Proposal	Description	Framework characteristics	Technologies	Privacy area
Securing MQTT by Blockchain-Based One-Time Password (OTP) Authentication [56]	Uses One-Time password generated on the MQTT Broker for authentication and authorisation of devices, done with smart contracts used in Ethereum Blockchains	Identity privacy; Interoperability; Query privacy; Security of data.	- MQTT; - Blockchain; - Smart contract for authentication and authorisation.	Data privacy and User privacy.
BlockChain: A Distributed Solution to Automotive Security and Privacy [63]	Proposes an architecture based on a Lightweight Scalable Blockchain (LSB), an optimised blockchain for IoT, to provide privacy and security for automotive applications. It aims to improve user privacy by anonymising the data and letting the user choose which data will be transmitted to service providers.	Access control; Openness, transparency, and a specified purpose; Temporal and location privacy; Identity privacy; Query privacy; Interoperability; Data minimization; Security of data.	- Blockchain.	Data privacy, User privacy, Location privacy and user-controlled privacy.
An IoT-Oriented Privacy-Preserving Publish/Subscribe Model Over Blockchains [76]	Proposes a privacy solution based on an Ethereum blockchain using a publish/subscriber architecture to function as a broker.	Access control; Identity privacy; Query privacy; Interoperability; Security of data.	- MQTT; - Blockchain	Data privacy and User privacy.
A blockchain-based location privacy-preserving crowdsensing system [95]	Implements a tiered architecture with public and private blockchains to enhance users' privacy, location, and transactions. Users data is distributed across multiple private blockchains to prevent re-identification attacks.	Openness, transparency, and a specified purpose; Identity privacy; Temporal and location privacy; Query privacy; Interoperability; Security of data.	- Tiered architecture with Public and Private blockchains with an agent; - Smart contracts for access control.	User privacy and Location privacy.
Blockchain for IoT Security and Privacy: The Case Study of a Smart Home [62]	Uses a private blockchain to store transaction made in the smart homes and for access control, and uses public blockchains to control the access to data in the smart home made by overlay devices (service providers, other devices/users).	Access control; Openness, transparency, and a specified purpose; Identity privacy; Query privacy; Interoperability; Security of data.	- Tiered architecture with Public and Private blockchains with the use of a smart home miner node; - Internal and external communications are control by the miner node, which checks the ACL stored in a block on both blockchains.	Data privacy and User privacy.
IoT data privacy via blockchains and IPFS [49]	It is an interconnected consortium blockchain network with multiple private blockchains (identified as private side-chains). Both types of blockchain have a smart contract that controls the access to IoT data. The private side-chains store encrypted IPFS file hashes of the IoT data that can be obtained and decrypted by authorised nodes, which obtain the data via an HTTP request.	Identity privacy; Temporal and location privacy; Query privacy; Interoperability; Security of data.	- Interconnected Public and Private blockchains with a validator node; - Smart contracts for access control; - IPFS for decentralised storage.	User privacy and data privacy.

In Table 4.2, is made an identification and classification of PETs that use blockchain technology. The classification of their characteristics is based on the privacy framework proposed in [81], the used technologies, and the private areas addressed, which were identified in section 3.1.

The presented proposals use blockchain to enhance the user's privacy and security and present data. Furthermore, based on the surveys ([50, 58, 66, 82]) the blockchain is identified as a promising solution to enhance the privacy and security of IoT applications, and more specifically automotive application ([58, 66]). However, blockchain alone is not private and secure enough without innovative solutions (e.g., smart contracts) and integrations (e.g., external decentralised storage system).

In some proposals of Table 4.2 ([49, 56, 62, 95]), smart contracts are used to perform certain tasks, such as accessing data and storing information. This solution can further improve the privacy and security of data stored in the blockchains. Although this mechanism needs to be implemented in private blockchains effective since public blockchains are open, any participant can access the information stored. Another proposed solution is using a tiered architecture with public and private blockchains, where both blockchains can transact between them. They are connected by a node that is a participant of both types of blockchains, which allows the interoperability of blockchains and improves the privacy and security of data by storing it in the private blockchains ([49]). This approach is described in [50] as the most promising solution for IoT data transfers.

4.1.3 Privacy proposals comparison

The identified privacy proposals use distinct mechanisms to provide privacy to users and data (location and other data types). Table 4.3 identifies the characteristics of each proposal, based on different aspects, with the purpose to summarise their functional characteristics and differentiate them.

By analysing Table 4.3, it is possible to state that most of the proposals have distinct characteristics. However, half of the blockchain-based proposals use a tiered blockchain architecture since it is identified as a promising solution for data transfers and blockchain interoperability [50]. Encryption of data is also integrated into some of the proposals. Most of them use asymmetric encryption, except for [62], which uses symmetric encryption. Access control is differently implemented in all proposals to enhance the security of the system, and therefore its privacy. User anonymity is a characteristic that is implemented by all blockchain-based proposals. By nature, blockchains provide some level of user anonymity since the identity of users is masked by blockchain addresses. Storage of data is only addressed in the proposals that use blockchain, although most of them relying on local or external storage solutions, except for in-chain storage in [95]. The storage is done on both public and private blockchains, although storing data on public blockchains is unsafe and does not provide privacy to data. Storing data in a private blockchain is more private and secure due to the permission-based nature of these kinds of blockchains. However, the immutability of blockchains prevent data management functionalities (e.g., delete or modify data).

Table 4.3: Characteristics of the identified privacy proposals.

Proposal	**Architecture**	**Confidentiality**	**Access control**	**Data storage**	**User anonymity**	**Location privacy**	**Validation strategy**
Enforcement of security policy rules for the Internet of Things [78]	MQTT broker with an integrated security toolkit (SecKit)	None	SecKit	None	Define a policy to obfuscate the username of the data owner	Obfuscate GPS data with a policy	Real implementation scenario
Secure mqtt for internet of things (IoT) [84]	Integrated ABE with MQTT	Asymmetric encryption(with ECC)	ABE	None	None	None	Real implementation scenario
Towards improving the Privacy in the MQTT Protocol [65]	OTP integrated in MQTT messages	None	AOTP	None	None	None	Real implementation scenario
Securing MQTT by Blockchain-Based OTP Authentication [56]	MQTT authentication using blockchain	None	Smart contract	None	Blockchain pseudo-anonymity	None	Security analysis without implementation
BlockChain: A Distributed Solution to Automotive Security and Privacy [63]	Blockchain	Asymmetric encryption	Overlay block managers	Local and cloud	Blockchain pseudo-anonymity	User-controlled location privacy	Security and privacy analysis without implementation
An IoT-Oriented Privacy-Preserving Publish/Subscribe Model Over Blockchains [76]	Blockchain with a publish/subscribe model	Asymmetric encryption	Authentication mechanism	In-chain	Blockchain pseudo-anonymity	None	Real implementation scenario
A blockchain-based location privacy-preserving crowdsensing system [95]	Tiered blockchain architecture	None	In private blockchains	In both types of blockchains	Blockchain pseudo-anonymity	Location obfuscated by an area	Emulation
Blockchain for IoT Security and Privacy: The Case Study of a Smart Home [62]	Tiered blockchain architecture	Symmetric encryption	ACL written on-chain	Local and cloud	Blockchain pseudo-anonymity	None	Simulation
IoT data privacy via blockchains and IPFS [49]	Tiered blockchain architecture	Asymmetric encryption	Smart contract	IPFS	Blockchain pseudo-anonymity	None	Real implementation scenario

The proposal [62] uses symmetric encryption and the Diffie-Hellman key exchange to share the secret key between nodes of the blockchain architecture. This approach is sufficient if used for end-to-end communications. However, it is not adequate to share data to various end nodes of the blockchain since many transactions and various individual secret keys would be needed to share data with each of the end nodes. Ideally, one symmetric key used by many nodes is preferable for this kind of scenarios to share data securely and efficiently.

4.2 Conclusions

In this chapter, the concept of privacy-oriented IoT applications was addressed by identifying their properties. A privacy framework is addressed in [81], it was used to distinguish the privacy proposals identified in section 4.1. The proposals were divided into two sections. One section is composed of proposals focused on improving the privacy and security of the MQTT protocol, and the others use blockchain-based solutions. Blockchain integration has been increasing, especially in privacy and security proposals for IoT, because of its inherent privacy and security features. However, some proposals have taken a step forward by connecting it to other blockchains and external entities (e.g., cloud and IPFS). Smart contracts have also revolutionised blockchains, since it allows the use of blockchains as a set of decentralised applications, and not only has a channel to exchange cryptocurrency.

Based on the identified PETs and their characteristics, our proposal will implement a blockchain-based architecture. In addition, some different approaches will be implemented further to improve the privacy of IoT applications with mobility. Our proposal is described in the next chapter.

Chapter 5

Privacy-Enhancing Proposal

In this chapter, our proposal is addressed, which consists of a blockchain-based system to enhance the privacy of automotive IoT applications. The proposal's requirements will be defined, as well as the architecture, the implementation strategy, and the evaluation strategy.

5.1 Application and requirements

In this section, an example of a use case for our system is provided and its main requirements.

5.1.1 Application

The main goal of this proposal is to enhance the privacy and security of automotive applications by storing the data and exchanging it securely and privately. More specifically, applications where smart vehicles share location data and telemetry data to service providers or other entities (e.g., crowdsensing). The in-vehicle sensor controller gathers this data to process the information related to events (e.g., accidents), road conditions (e.g., traffic jams, debris on the road), best routes and much more. Then, it sends the data to be stored and later exchanged to authorised users.

We consider an application in the context in which smart vehicle owners (vehicle users) wish to share data about accidents, traffic jams, or other events to GPS applications to alert authorities or for research purposes. Each one of these kinds of data has its priority, required privacy and end applications. The vehicle user has the right to define who can access each one of these types of data by specifying the blockchain addresses of the nodes in the public blockchain allowed to access the data by sending the command: *define "type of data" "ACL"*, to the MQTT broker. These blockchain addresses can be obtained through an application deployed in-chain in a smart contract or another external application. However, this mechanism will not be addressed or implemented in this version of the system. Then, the broker executes the function *defineACL(owner, dataInfo, acl)* of the PBVU's smart contract to store the defined ACL for that specific vehicle user's data type. The smart contract stores this information in a dynamic array of structures, where each cell of the array represents a vehicle user. Finally, a symmetric key is generated in the sensor controller and exchanged securely, using events and encrypted with asymmetric cryptography, to the public blockchain nodes of the ACL.

When the sensors of the car detect an event, the smart vehicle's sensor controller sends the command *publish "data type" "data about event"* to the MQTT broker. The data is sent encrypted with the symmetric key, previously exchanged with the authorised nodes. The broker stores the data in Storj (decentralised storage system), and a reference to access the data is generated. Then, the broker executes the smart contract *addData(owner, dataInfo, serial, ID)* function, which stores the reference in the smart contract. Subsequently, the log information of the data is transmitted by the smart contract to the smart contract proxy to broadcast it as a log event by the smart contract of the public blockchain. The public nodes can fetch the log information about the newly added data and make a decision.

Next, if the authorised nodes want to request the data, they execute the *getPrivData(ID)* function of the public blockchain smart contract. The request is forwarded to the smart contract proxy, which executes the smart contract of the PBVU to perform access control. If the node is authorised, the serial is retrieved and broadcasted to the public blockchain as an event. Even if other nodes try to parse it, they cannot decrypt the data without the symmetric key.

With the data, the node can use it for their intended application or store in their systems private blockchains (e.g., private blockchains, storage systems). These nodes can be owned by GPS applications, authorities or research entities. For example, some end application can be: display information on a GPS system (e.g., Google Maps or Waze), alert authorities about an accident, or to be used for research purposes to improve roads.

This proposal assures vehicle user privacy by decoupling the data generated with information about the owner, providing anonymity of the user with the blockchain addresses, and preventing transactions' traceability. The location and data remain private and secure by using decentralised access control mechanisms and symmetric cryptography throughout transport and when stored.

5.1.2 Requirements

Blockchain

To enhance the privacy and security of automotive applications, we propose using a tiered architecture of blockchains, particularly a public blockchain connected to multiple private blockchains. Only one public blockchain is used, where all parties of the system can participate since the access to the public blockchains is permissionless. In terms of private blockchains, the number of present blockchains of this type in the system is undefined. It depends on the number of participating public blockchain nodes participating in private blockchains (agents). However, at least one private blockchain is always present, the PBVU, where their generated data is stored and controlled by a smart contract. This architecture increases the privacy and security of data and the performance and scalability of the blockchains by implementing various blockchains with distinct goals and functionalities.

In the following Tables 5.1, 5.2 and 5.3, the requirements related to the blockchain technology implemented in our proposal are going to be specified.

Table 5.1: Blockchain functional requirements.

Name	**Description**
Intermediation between blockchains	Allow executing function calls between smart contracts from different blockchains to handle requests that require cross-blockchain communications (e.g., data requests and key exchanges).
Decentralised storage	Provide off-chain data storage to the vehicle users generated data in a decentralised system, which can be accessed in-chain via a data reference (serial).
Data management	The vehicular users can manage their data. They can decide which data is sent, who can access it, and access or delete/modify data previously sent.
Access control	Provide access control to vehicle users generated data, which is stored as a reference in the PBVU's smart contract. The ACL of each data type is also stored in the smart contract.
Accountability	The Smart contract generates log information (ID, data type and timestamp) about new data added in the PBVU, which are broadcasted as an event in the public blockchain. This log event informs the public blockchain nodes that new data is available and identifies it by its ID and data type.
Flood prevention	Blockchains, specifically public blockchains, use cryptocurrency as a currency to be transferred between nodes but is also used as an incentive to keep the blockchain secure (reward systems: staking [55]). However, in this system, cryptocurrency is also intended to be used as a flood prevention method by imposing fees on data access requests made in the public blockchain. It has the goal to prevent malicious nodes from flooding the blockchain with requests.

Table 5.2: Blockchain privacy and security requirements.

Name	**Description**
User controlled privacy	The user can define which data is sent and which end nodes can access it.
User anonymity	Blockchain accounts provide enhanced privacy to the users since the address of each account is not associated with any personal information. Therefore the system does not store sensitive information about the user.
Location and data privacy	Data sent to authorised nodes is completely anonymous, which prevents user tracking and profiling attacks.
Data confidentiality	Data, when it is stored and in transit, is always encrypted with a symmetric key.

Table 5.3: Blockchain non-functional requirements.

Name	**Description**
Performance	Improve the performance of the system by distributing the workload through multiple blockchains to prevent congestion. Each blockchain has its functionalities and goals. Together they form a complete system. A private blockchain will be used to store the data, and the public blockchain to access it.
Blockchain size-reduction	Improve the scalability of the blockchains through the distribution of requests into different blockchains with different purposes. Therefore, minimise the size growth of the blockchains to reduce the amount of storage resources needed.

Our proposal has merged some aspects proposed in blockchain Privacy-Enhancing Technology (PET)s (chapter 4.1.2), such as blockchains to improve the security and privacy of data, tiered architecture of blockchains, connected by the agent nodes for connectivity, performance and scalability, and smart contracts for access control. However, it offers

some distinctive approaches. The main one is connecting the blockchain of vehicle users (where data is stored) and the public blockchain using a smart contract proxy, responsible for handling requests made by smart contracts from different blockchains.

Furthermore, the data is encrypted using symmetric encryption instead of the classical asymmetric encryption used in other systems. It intends to encrypt the data at the source and offers the possibility to be decrypted by multiple authorised nodes, eliminating the need to send the encrypted data multiple times.

MQTT

To improve the performance and the resilience of IoT applications with mobility, a lightweight messaging broker (MQTT) is used in conjunction with his QoS and durable connection features. Despite the security measures of MQTT not being ideal for constrained devices, data confidentiality, authentication and authorisation will be implemented to provide more security to MQTT communications.

The following tables contain the requirements related to MQTT in this system.

Table 5.4: MQTT functional requirements.

Name	Description
Request processing	The MQTT broker is responsible for receiving messages from the vehicle users, executing the smart contract of the PBVU, or performing other action.

Table 5.5: MQTT security requirements.

Name	Description
Authentication	The sensor controllers need to authenticate themselves in order to connect to the MQTT broker.
Authorisation	Provide permissions to sensor controllers to perform certain tasks (e.g., subscribe and publish or neither one) in certain topics to prevent eavesdropping or tampering data sent by other sensor controllers.
Data confidentiality	The MQTT communications between the sensor controller and the broker must be encrypted to prevent eavesdropping and man-in-the-middle attacks.

Table 5.6: MQTT non-functional requirements.

Name	Description
Interoperability	Allow constrained devices that do not support blockchain technology to communicate with this blockchain-based system.
Bandwidth consumption reduction	MQTT uses small packets, which improves the performance of the communications by consuming less bandwidth.

The MQTT broker in this proposal allows the smart vehicles to transmit data with fewer resource requirements and more resilience. However, the new concept implemented in this proposal is that the MQTT broker is a member of the PBVU, which has the responsibility to make function calls based on the commands of the messages sent by the sensor controller. The sensor controller never communicates directly with the PBVU.

5.2 Motivation

Our privacy-preserving proposal is based on the previously identified PETs (Table 4.3) and other articles present in the literature. However, our proposal will provide some distinct characteristics to improve certain limitations of the identified proposals.

In [69], MQTT is used to improve the communication quality between smart vehicles, as we considered for our proposal. Applications with mobility, such as the connectivity of smart vehicles to applications in the cloud, can be unreliable and unstable. MQTT can improve the communication of these types of applications by using its QoS mechanisms, durable connection feature and by providing a lightweight messaging broker. These features of MQTT improve the resilience and performance of these IoT applications with mobility by preventing the loss of messages and connection with MQTT and reduces the amount of necessary computational and network resources.

An MQTT broker will be used as a communication middleware between the IoT device and the blockchain in our proposal. It is similar to [56], although, in the proposal, blockchain is used to improve the security of MQTT authentication and authorisation mechanisms. It is used to provide a lightweight communication protocol and to provide resilience to communications. The broker is also responsible for making transactions in the name of the IoT device based on the messages published by it. Unlike in [62] where each IoT devices can communicate directly with the public blockchain, and vice-versa, which could be a security threat if the access control mechanism used in this proposal does not function as designed or if the ACL is poorly configured. It is also expected to remove some computational overhead from the IoT devices, which will be verified through experimental testing.

In the [95], [62] and [49] proposals are used two connected blockchains (tiered architecture of blockchains). Each kind of blockchain has different goals and functionalities. They are connected to take advantage of their benefits. The blockchains in these proposals are connected by a blockchain node that responds to requests to access data or other types of requests made by the nodes.

Our proposal uses a similar approach, but the Private Blockchain of Vehicle Users (PBVU) communicates with the public blockchain with the use of a smart contract proxy. It is responsible for handling certain requests generated by smart contracts from both blockchains. In contrast to other proposals, smart contracts are the only entities in the blockchain that can make requests to the proxy. Moreover, other private blockchains or systems (owned by enterprises or other kinds of entities) can be connected to the public blockchain by a node, called agent (similar to [95]).

A tiered architecture with public and private blockchains will allow participants to make transactions between different kinds of blockchains, enhancing the interoperability and scalability of blockchains. Each blockchain has its purposes. Therefore, the workload is distributed through multiple blockchains instead of only residing in one blockchain, enhancing the system's scalability.

Similarly to [49], data in our system is stored off-chain and is never stored in the public blockchain for privacy reasons, unlike in [62]. Public blockchains are transparent. Thus, every node can access the transactions of the blockchain. For this reason, it is only stored log information of the data in the smart contracts.

Storj [40] is used as the off-chain storage solution, which allows eliminating data, unlike IPFS that does not provide a confirmation that data was deleted [25]. It also provides more security and reliability features than IPFS [49] or other cloud storage solutions [62].

Similarly to [49], in the blockchain, it is only stored a reference (serialised access string) of the data, which can be used to access the data.

In terms of data confidentiality, our proposal uses symmetric encryption, just like in [62], to encrypt the transmitted data in order to be only decrypted by the authorised nodes. However, in [62], the data is encrypted for end-to-end transmissions, which allows them to use the Diffie-Hellman key exchange to share the key. This method can be quite cumbersome to exchange keys between a high number of devices. In our system, the symmetric key is securely exchanged using asymmetric cryptography before the vehicle user publishes any data. The public and private keys used in asymmetric encryption are the ones created to generate the wallets of each blockchain node. The encrypted key will be exchanged through the public blockchain as an event addressed to the destined public node. The public nodes can fetch this event. However, only the authorised nodes can use their private key to decipher it. Only one secret key will be used to encrypt each data type of each vehicle user, which reduces the number of transactions made to transfer data, therefore improving the scalability of the blockchain by using it more efficiently.

The access control mechanism is similar to the one proposed in [49], which is performed with a smart contract. It contains a list of the blockchain addresses of authorised nodes from the public blockchain. Although, the vehicle users are responsible for the content of the ACL. Similar to [63], our proposal offers user-controlled privacy by design since the user has the right to choose who can access his data. This ACL is used per data type, and the authorised nodes are the recipients of the exchanged symmetric keys.

User anonymity, much like in other blockchain-based proposals, is provided by default with blockchain account creation. The blockchain addresses that represent each account do not correlate with the users' sensitive information.

Location privacy and data privacy are implemented in some proposals via mechanisms of access policies, user-controlled privacy [63] and location obfuscation ([78, 95]). In our proposal, this is achieved by implementing the first two mentioned mechanisms (access policies and user-controlled privacy). Although, the data and location information are further anonymised by removing any information associated with its owner (e.g., blockchain address) when providing the data to the authorised public nodes. This mechanism provides anonymity by design, which prevents any linking attack, consequently removing any chance of location tracking to be performed.

In [95], the smart contract imposes a deposit of cryptocurrency from both ends (data generator and service provider) as an assurance. If both parties comply with the predefined clauses, the data generator receives the payment from the service provider, and the service provider receives the data. Our proposal also imposes a payment in cryptocurrency (fee) by the public blockchain smart contract when a node requests data access. This fee prevents a flood of requests from being made by unauthorised nodes since it becomes expensive to flood the system, which could prevent it from Denial-of-Service (DoS) attacks and could save computational resources.

Beyond the identified solutions and modifications proposed to be implemented in our proposal, one different solution will be implemented in the proposal. It uses a smart contract proxy, which has a similar concept to a proxy server. However, it is used as an intermediary for requests done by smart contracts of both blockchain types. Data requesters can use this mechanism to request data from a private blockchain. They call a function from the public blockchain's smart contract, and it is forwarded to the private blockchain's smart contract.

In sum, our proposal uses some features of the architecture of the proposals identified in Table 4.2 that provide privacy and security to data but also proposes other distinct approaches to improve the privacy and security of automotive applications.

5.3 Architecture

The system's architecture allows the exchange of data privately through blockchains. As illustrated in Figure 5.1, the system is composed of the tiered architecture of blockchains connected by bridge devices. Data generated by smart vehicles is stored off-chain in Storj and can be accessed through a serialised access stored in the private blockchain, wherein the smart contracts manage access control.

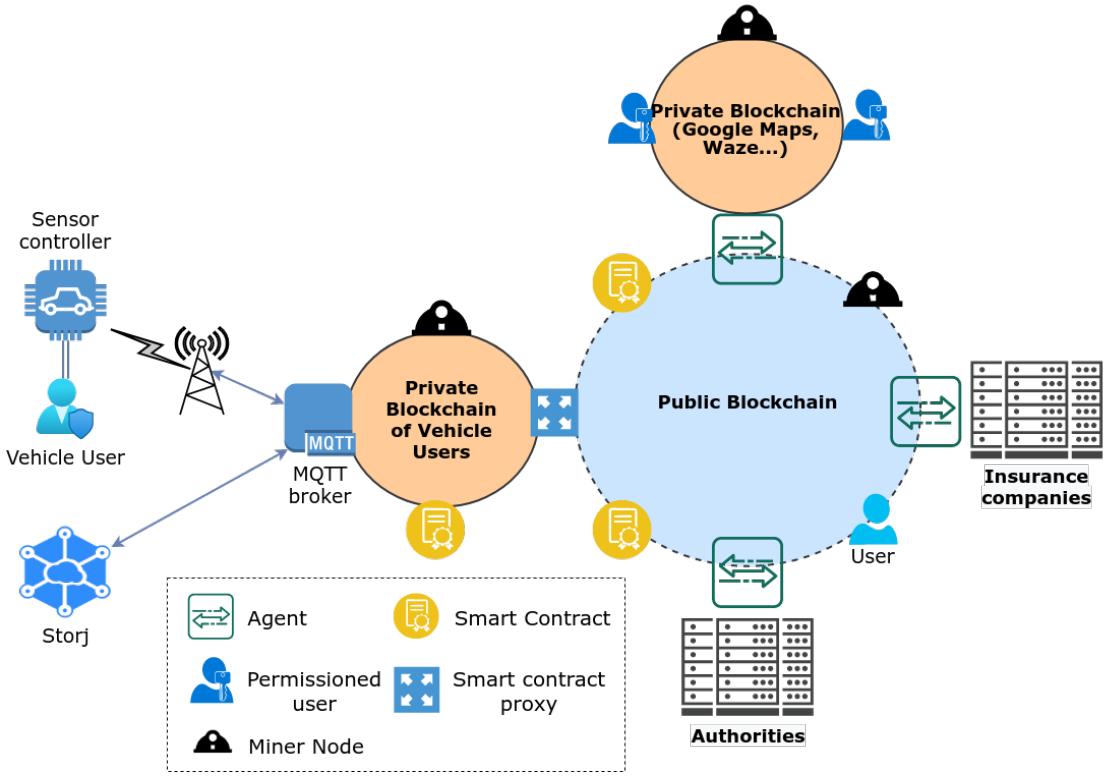

Figure 5.1: Architecture of the proposal solution.

5.3.1 Architecture components

- **Public Blockchain:** Open and permissionless blockchain where every node can participate. In this blockchain, it is only stored log information (ID, data type and timestamp) of the data published by the vehicle users as events. It functions as a public log ledger and access request relay. Any node of the public blockchain can request access to data addressed to the public blockchain's smart contract.

- **Private Blockchains:** Closed and permission-based blockchains where only invited and authorised nodes can participate. The private blockchain can be of any permission-based blockchain. However, it is managed by the participant entity. These blockchains can be used by companies, government entities, or other entities to access

the data produced by vehicle users. They can provide granular control of the nodes, are faster, more efficient, and more cost-effective than public blockchains because of the reduced number of nodes and reduced chain size.

- **Private Blockchain of Vehicle Users (PBVU):** In this private blockchain, only MQTT brokers, the smart contract proxies, and miner nodes participate. Its smart contract is used to store data's serials and ACLs and exchange information to the public blockchain. In this blockchain, cryptocurrency is not an aspect taken into account since it is not computationally intensive to mine blocks in private blockchains, and it is assumed that each node is trusted since it is a permission-based system.

- **Storj [40]:** Decentralised and S3-compatible storage platform (uses the Amazon S3 Application Programming Interface [39]). The files are encrypted using the AES-256-GCM algorithm, by default, with a locally stored private key. The data is stored as 80 fragments of data distributed through 80 servers in a globally decentralised network, which offers enhanced security (prevents data breaches) and resilience to data loss (only 29 pieces are needed to reconstruct the data). The vehicle users data will be stored in this system. Moreover, to access the data, a serialised access string is generated for each data entry and is stored in the PBVU's smart contract.

- **Agent:** It is a trusted full node of the public blockchain. Its responsibility is to bridge the blockchain and the private infrastructure of the entity (e.g., servers or a private blockchain). They request data access to the public blockchain smart contract, and if they are authorised by the owner of the data (vehicle user), they receive it. Then, they use it for their intended purpose. The agents can also function as validators of blocks in the public blockchain. These nodes should be devices with some computational resources, especially storage capacity to store the state copies of all blockchains they participate. In addition, these nodes are owned and managed by the entities that request the data.

- **Smart Contract Proxy:** It is a node that is connected to the PBVU and the public blockchain. It has software in execution that keeps listening for events made from smart contracts addressed to him. It contains a list of the blockchain addresses of the smart contracts from both blockchains, which allows it to only accept requests made from these smart contracts. Similar to MQTT messages, the events also have commands to instruct the proxy to execute certain tasks. They are sent as dictionaries with three keys: "command" (identifies the command), "dataInfo" (an array of the data itself) and "destinationAddr" (optional: used to identify the destined blockchain address). The commands are:

  - *createTransaction "data info" "destination"*, this commands the proxy node to create a transaction on the public blockchain with the information provided, and it could be addressed to a node or not (the destination address is optional). Usually, it is used by the PBVU's smart contract.
  - *createLog "data info"*, this commands the proxy node to call the function *setLog(dataInfo)* of the public blockchain's smart contract, which generates an event with the log information of the newly added data.
  - *getData "data info" "requester"*, this command is generated by the public blockchain's smart contracts to forward data requests from public nodes. The ID of the data and the node requester's blockchain address are sent for access control purposes. When received, the proxy executes the function *getData(ID, reqAddr)* of the PBVU's smart contract.

- **Smart Contracts:** They are deployed in the blockchains with a specific blockchain address and are responsible for executing a certain number of programmed functions. The functions available to be called by the blockchain nodes are:

  1. ***defineACL***, receives as a parameter the owner address, the type of data and the list of blockchain address of the public blockchain nodes authorised to receive this data type, which each vehicle user generates. When executed, it creates a transaction and stores the list of authorised blockchain addresses in the PBVU.
  2. ***exchangeKey***, receives the symmetric key encrypted with a public key of an authorised node as a parameter. This function is responsible to request the creation of an event on the public blockchain addressed to the node allowed to decrypt the symmetric key. A request is sent to the smart contract proxy to create this event, which executes the *setKey* function to generate it in the public blockchain.
  3. ***addData***, receives the data information (owner address, data type and ID) and the serial of the data. Since the MQTT broker previously authenticated the blockchain address of the owner, the smart contract does not need to check that he is authorised to add data. Then, it proceeds to store the data, which generates a transaction inside the PBVU and creates an event to request the proxy node to generate log information about the data in the public blockchain.
  4. ***setLog***, receives the information about newly stored data, sent by the proxy nodes, to create a log event. The public blockchain nodes use the logs events to fetch the data's information, including its ID, to make access requests.
  5. ***setKey***, is responsible for generating the event to broadcast the encrypted symmetric key in the public blockchain to be parsed by the destined node. It receives the address of the destined public node and the encrypted symmetric key.
  6. ***getPrivData***, receives the ID of the data requested by the public blockchain node as a parameter. This function also receives a specific amount of cryptocurrency as a fee to pay for the subsequent transaction needed to request the data and prevent floods of requests from public nodes. Then, a request is made to the proxy as an event to request the data to the PBVU's smart contract.
  7. ***getData***, it is executed by the smart contract proxy, which receives the ID of the data and the blockchain address of the requester as a parameter. Then, suppose the node is authorised to receive the data. In that case, its serial is sent back to the proxy, which forwards it to the public blockchain's smart contract by executing the *setNewDataLog* function.
  8. ***setNewDataLog***, it sends the *get data* request response as an event to the public blockchain addressed to the data requester.

  There are two types of smart contracts in this system: PBVU's smart contract and public blockchain smart contract. The first one is responsible for storing the serials of the vehicle users' data, exchanging secret keys, managing access control, and responding to data requests from the public blockchain (it uses function number 1,2,3 and 7). The other receives access requests from public nodes (users and agents) and requests to broadcast events (it uses function number 4, 5, 6 and 8).

- **Vehicle Users:** They communicate with the MQTT broker to send their requests. Each user has an Ethereum account that posses a unique address. This address is used for identification purposes in the PBVU and authentication in the broker. It uses the sensor controller to publish messages, which sends commands to the MQTT

broker. Each user has their MQTT topic to publish messages and cannot publish or subscribe to other topics. These commands are sent as dictionaries with three keys: "command" (identifies the command), "dataInfo" (an array used to identify data types and blockchain address of a node or the owner) and "value" (used for encrypted data or other information). The commands used are:

  - *defineACL "data type" ACL*, this commands the MQTT broker to define the list of authorised public nodes that can receive the type of data specified. It orders the broker to execute the function *defineACL(owner, dataInfo, acl)* in the PBVU's smart contract.
  - *key "blockchain address of the authorised node" "encrypted key"*, this command is used to exchange the secret key with an authorised public node. The MQTT broker executes the function *exchangeKey(destNode, encKey)*, which results in the creation of an event in the public blockchain addressed to the authorised node, with the encrypted symmetric key.
  - *publish "data info" "data"*, commands the MQTT broker to store the data in Storj and execute the function *addData(owner, dataInfo, serial, ID)* of the smart contract to store the data serialised access string on the PBVU.
  - *access*, commands the MQTT broker to retrieve all the previously sent data. It displays all information related to the data, including its ID and timestamps.
  - *delete "data ID list"*, orders the MQTT broker to delete a specific data entry. The ID of the data is obtained by the *access* command.
  - *modify "data ID" "data"*, instructs the MQTT broker to modify a specific data entry and replace it with the new data sent. The URL stored by the smart contract does not need to be updated since the stored object is the same, only the content has changed.

- **Sensor controller:** It is a device present on the smart vehicle that receives the data from the car sensors and is responsible for transmitting it to the MQTT broker when an event or other relevant information is identified. Various data types can be sent. Each has different kinds of information, which has different application purposes and different destination nodes to be sent. This device also hosts an application where the user can define the access list for each data type he sends and manage his data. This device does not need many computational resources, e.g., a Raspberry pi, but should be a device capable of operating an Operating System (OS), and some applications in parallel.

- **Users:** They are private or public blockchain nodes (full or light nodes). A unique key pair is created when their blockchain user account is created. They can perform the same actions as the agent nodes. Nevertheless, they do not necessarily represent an entity. Thus, they can be users of an application.

- **Miner:** Their responsibility is creating blocks on the blockchain in which they participate.

- **MQTT Broker:** Full node of the PBVU, it receives the publish messages from vehicle users and, depending on the command, executes the corresponding function of the blockchain's smart contract or other action. It also authenticates vehicle users and imposes restrictions on the topics that they can interact. Furthermore, the communications are secured using confidentiality mechanisms.

5.3.2 Functionalities

- Privacy and security:
  - **Data erasure**, it is a GDPR requirement (Article 17), which allows the user to erase or rectify his data. This feature is provided through the user application, which relays the request from the sensor controller to the MQTT broker to make changes or erase the specified data.
  - **User defined access control**, the vehicle user has the right to choose, via the sensor controller application, the public nodes that can receive his data by data type using their blockchain address. This information is stored in the PBVU's smart contract. This smart contract is responsible for performing the access control. This access control layer provides user-controlled privacy by design, which increases the privacy and security of data. Moreover, since smart contracts are tamper resilient (the code remains unchanged) and decentralised, the ACLs cannot be modified by malicious nodes.
  - **User anonymity**, blockchains, by default, create a blockchain address using the account's private key previously created. These addresses are not associated with any information about the user. Therefore this system does not store any user's private information.
  - **Location and data privacy**, location data and other types of data stored are anonymous, which means they are not linked with any user. The events used to notify and exchange the data does not contain any information about the user. Only the data itself can contain sensitive information in specific data types that require sensitive information about the vehicle user (e.g., the user was involved in an accident and provides sensitive information to certified and authorised insurance and authority nodes). This mechanism offers anonymity by design.
  - **Data confidentiality**, data is always encrypted with symmetric cryptography, which can only be decrypted by authorised nodes using the pre-shared symmetric key. The symmetric key is distributed to the authorised nodes with asymmetric cryptography. Thus, an event has to be created for each authorised nodes of each data type. This process can be slow in the first phase of key exchange. However, it is more scalable for multiple end nodes in the long run since the data is encrypted with a symmetric key.
  - **Access control immutability**, by design, blocks on the blockchain cannot be altered since the blocks are connected by their hash and stored on multiple distributed and decentralised nodes. Therefore, the ACLs defined by the nodes cannot be tampered by malicious nodes.
- Performance, scalability and interoperability:
  - **Interoperability between systems**, different types of blockchains can communicate with each other, which allows the connection of blockchains with different purposes to work as a unified system. Additionally, other systems can also be integrated into this architecture, such as IoT devices that support MQTT and cloud computing.
  - **Less overhead in the public blockchain**, by using private blockchains to handle requests from the sensor controller and other devices, fewer requests are made in the public blockchain, which reduces the congestion and the blockchain growth. Therefore, integrating new nodes in the public blockchain takes less time, and more bandwidth is available in the blockchain.

- **Flood requests prevention**, the smart contract of the public blockchain imposes a fee to perform the *getData* function. This feature prevents unauthorised nodes, or even authorised ones, from flooding the public smart contract, which reduces the congestion of the system and can prevent DoS attacks. In the future, the fees paid by these nodes can be used to pay the vehicle users for sharing their data.
- **Lightweight messaging system**, MQTT offers a lightweight solution for message exchanges made by constrained devices addressed to the blockchain system.
- **Resilience in communications between constrained devices and the system**, MQTT's QoS mechanisms assure that messages are received with more reliability by checking if the data is received and resends it if necessary. Durable connections ensure that when the device disconnects the MQTT broker, his subscriptions are saved, and any QoS message is stored until the device reconnects.

5.4 Implementation strategy

The implementation of this architecture will be focused on the privacy, security, and scalability (computational resource usage) aspects of the system. Thus, the application of this system in a real scenario will not be implemented. It will only be addressed with high-level exemplifications. How the data is handled and used in the end applications is the responsibility of the participating entities.

The main components of this system to be implemented are:

- **Tiered architecture of blockchains**, it is the implementation of a communication link between different blockchains using a smart contract proxy, which allows cross-blockchain communications between smart contracts. In this scenario, only one PBVU and one public blockchain will be integrated.
- **Smart contract proxy**, device and software that is responsible for cross-blockchain request handling.
- **Smart contracts**, blockchain entity containing all the previously mentioned functions (section 5.3.1).
- **Sensor controller application**, responsible for publishing and encrypting data and selecting the list of authorised users that can access the data.
- **MQTT broker**, device and software that handles the commands received by the MQTT broker node, published by the sensor controller, which executes the smart contract functions and manages the data stored in Storj based on the commands.

5.4.1 Implementation scenario

The whole system will be implemented on a physical device with the aid of a VMware hypervisor. The nodes will be allocated in various Virtual Machines (VM) to represent the various types of nodes of the system (smart contract proxy, agent and sensor controller). The Ropsten Ethereum testnet [45] will be used as the public blockchain of the architecture, an Ethereum private blockchain will be created to be used as the PBVU. One public blockchain and one PBVU will be implemented.

5.4.2 Requirements implementation

In this section, the implementation methodology of the requirements is described.

Blockchain

Tables 5.7,5.9 and 5.8, contain the requirements and their respective implementation strategies.

Table 5.7: Blockchain functional requirements implementation.

Name	**Implementation**
Intermediation between blockchains	The node that links operations from smart contracts of both types of blockchains is the smart contract proxy. The smart contracts will handle function calls made by blockchain nodes, which, in most cases, requires cross-chain operations. It is the job of the proxy to exchange or handle these requests and responses between blockchains.
Decentralised storage	Data generated by the vehicle users will be stored in the decentralised storage system Storj by the MQTT broker. Then, the smart contract of the PBVU will store the generated serial of the stored data to be accessed by an authorised node of the public blockchain.
Data management	In the sensor controller application, the user can specify the ACL of each type of data, and he can also access, delete and modify his data. These requests related to his data are sent as commands by the sensor controller to the MQTT broker, responsible for handling his requests.
Access control	The ACL defined by the user are sent to the PBVU's smart contract through the MQTT broker, which preserves this information in-chain. Each user has a slot in the smart contract where all his corresponding privacy preferences are stored. Then, when a data request is performed, the ACLs are used to check who can receive the data.
Accountability	When the smart contract receives new data, it requests the proxy to generate an event in the public blockchain with log information (ID and data type). An ID is generated by the MQTT broker when it stores the data in Storj, which is used to identify the data when a node desires to request it. However, the log information does not identify the user that generated the data.
Flood prevention	The function call *getPrivData(ID)* will impose a fee to be paid. The goal is to pay the subsequent transactions and to prevent flood requests from any node (authorised or not authorised) with malicious purposes (DoS or for exploit attempts of trying to get data illegitimately).

Table 5.8: Blockchain non-functional requirements implementation.

Name	**Implementation**
Performance	Using private blockchains to handle certain requests improves the system's performance since they are faster at validating transactions due to their simpler consensus protocol.
Blockchain size-reduction	By storing the serials in PBVU and handling other requests, the size of the public blockchain does not increase as much compared to classical public blockchain approaches with only one chain. Thus, the scalability of the system is improved.

Table 5.9: Blockchain security requirements implementation.

Name	Implementation
User controlled privacy	The user defines the ACL for each data type used to verify which nodes are authorised to receive his data. This ACL will be sent as a command to the MQTT broker to execute the smart contract to store it and associate it to the user's privacy preferences.
User anonymity	By default, blockchains calculate the blockchain address of blockchain nodes using a one-way cryptographic hash (typically SHA256) with the node's pre-generated public key. The private key is a random and unique 256-bit number generated for each blockchain node used to sign transactions (authenticity purposes) and generate the public key with a cryptographic algorithm. Using these blockchain addresses to identify users can improve the anonymity of users in the blockchain transactions since they do not correlate with the real identity of the users. Furthermore, when data is sent to authorised public blockchain nodes (agents or users), any link between the data provided and the vehicle user that generated it is not included.
Location and data privacy	When the smart contract sends data to the public blockchain nodes, only data is included. Other information is not provided about the owner. However, the data itself can contain sensitive information, but the user is responsible for selecting who is authorised to access his data that contains personal information.
Data confidentiality	The vehicle user exchanges a symmetric key, previously generated by him, to the authorised public blockchain users of his ACL. The symmetric key, when in exchange, is encrypted with the authorised users public key, which is sent as an event by the smart contract in the public blockchain addressed to the public node. The allowed nodes can decrypt it with their private key and store it to decrypt future data.

MQTT

Similar to the previous section, Tables 5.10, 5.11 and 5.12, also contain the requirements and their respective implementation strategies.

Table 5.10: MQTT functional requirements implementation.

Name	Implementation
Request processing	The MQTT broker is implemented as a member of PBVU to make function calls addressed to its smart contract.

Table 5.11: MQTT security requirements implementation.

Name	Implementation
Authentication	The plugin "File RBAC Extension" will be used on the HiveMQ broker to authenticate users. This plugin uses an XML file that has a list of usernames and passwords to authenticate. The usernames will correspond to the vehicle users' blockchain address to identify the sender of the message. The list of users is pre-registered. However, the application to perform the account creation will not be implemented since the core privacy-enhancing functionalities are the main focus of this implementation phase.
Authorisation	"File RBAC Extension" plugin will also be used to provide fine-grained access control on a topic-filter level. Each user will have its own topics where they can publish data.
Data confidentiality	Transport Layer Security (TLS) will be implemented to provide a secure communication channel between the sensor controller and the broker, despite being considered resource-intensive for IoT devices.

Table 5.12: MQTT non-functional requirements implementation.

Name	Implementation
Interoperability	Using the MQTT broker to handle the requests from the sensor controllers offers interoperability with blockchains to devices that do not support blockchain technology since the interaction with the blockchain is decoupled from the device. The only restriction is that they support MQTT communications. Nevertheless, nowadays, it is supported by a wide range of constrained and non constrained devices.
Bandwidth consumption reduction	By default, MQTT is a protocol that uses small headers (2 bytes) to receive and transmit messages generated by devices. It receives published data from the sensor controller (present in the smart vehicle), which commands the MQTT to perform certain smart contract functions or other data management related tasks.

5.4.3 System's processes

In this section, the way the processes work will be explained by illustrating the interactions between the architecture's components.

Access control list definition

In Figure 5.2, it is illustrated how the vehicle user defines the list of users that can receive his data. The sensor controller starts by publishing a message with the command: *defineACL "data type" "ACL"*. The *data info* contains the data type and that the ACL encompasses, and the *ACL* is a list of public nodes addresses. This message commands the MQTT broker to execute the function *defineACL(owner, dataInfo, acl)* of the smart contract of the PBVU. Then, the function stores the list of addresses in the user's privacy preferences.

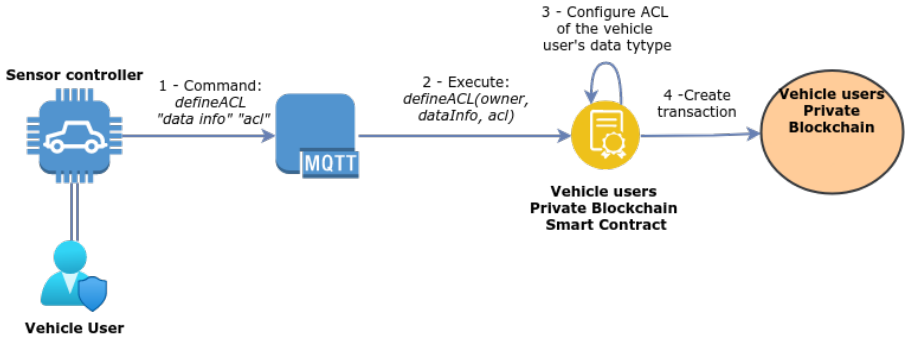

Figure 5.2: Vehicle user's ACL definition.

Key exchange

After the ACL is defined, a symmetric key is generated to encrypt the following data exchanged by the sensor controller. It allows multiple authorised nodes to decrypt it with the same key, preventing multiple data encryptions for each node. Therefore in Figure 5.3, the sensor controller begins by generating a unique 256-bit symmetric key to be used on specific data type transactions. Then, he encrypts it individually with the public keys of all the authorised nodes present on the ACL of the specific data type, to be later decrypted with their private keys. The public keys could be obtained from another application

(not addressed in this proposal). It is the responsibility of the vehicle user to choose the authorised nodes. The sensor controller proceeds to publish a message with the command: *key "blockchain address" "encryptedKey"*, it sends the blockchain address of the authorised node and the key encrypted with his public key. This commands the MQTT broker to execute the function *exchangeKey(destNode, encKey)*. When executed, the smart contract creates an event to request the proxy to create a transaction on the public blockchain addressed to the authorised node. Additionally, this process is executed every time a new ACL is defined by the vehicle user, with the purpose to renew the symmetric key when the ACL is modified.

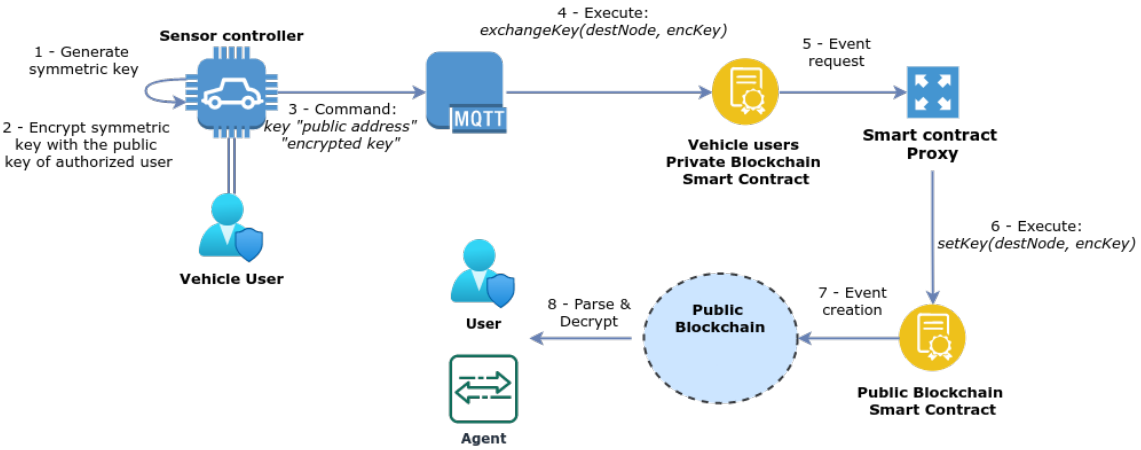

Figure 5.3: Symmetric key exchange process to authorised users.

Publishing and storing data

Figure 5.4 illustrates the data sharing process. The sensor controller begins by encrypting the data with the symmetric key, which was previously distributed in 5.3 to the authorised users. Then, it publishes a message with the command: *publish "data type" "encrypted data"*. This command instructs the MQTT broker to store the data in Storj and generate the serialised access string. Next, the smart contract's function *addData(owner, dataInfo, serial, ID)* is executed by the broker with the respective parameters. The *ID* is a Universally unique identifier (UUID) generated by the broker for each data entry. When the function is executed, the smart contract stores the serial and, consequently, generates another event to request the proxy to execute the function *setLog(dataInfo)* of the public blockchain's smart contract to generate a log event.

Data access request

In this interaction (Figure 5.5), a public node (agent or user) of the public blockchain requests access to data inside the PBVU, which was logged by the public blockchain smart contract. To perform this request, it executes the *getPrivData(ID)* function of the smart contract, where he specifies the ID of the data and sends the amount of cryptocurrency needed for the action. Otherwise, the request is denied. Next, the smart contract generates an event to request the data serial to the smart contract proxy. The proxy executes the *getData(reqAddr,ID)* function of the smart contract in the PBVU, which performs access control on the requester. If the node is authorised, the serial of the data is retrieved to the proxy, which executes the *setNewDataLog(requester, response)* function of the public

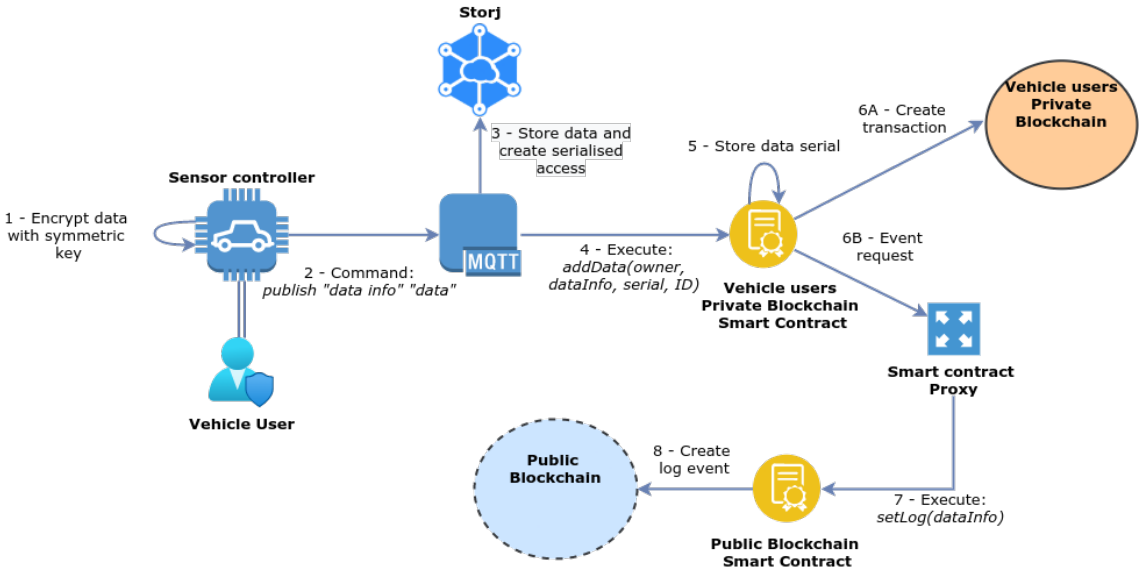

Figure 5.4: Data published by a sensor controller to be stored in Storj and in the blockchain system.

blockchain's smart contract. This function retrieves the serial to the requester node as an event. Finally, the requester can decrypt by using the pre-shared symmetric key.

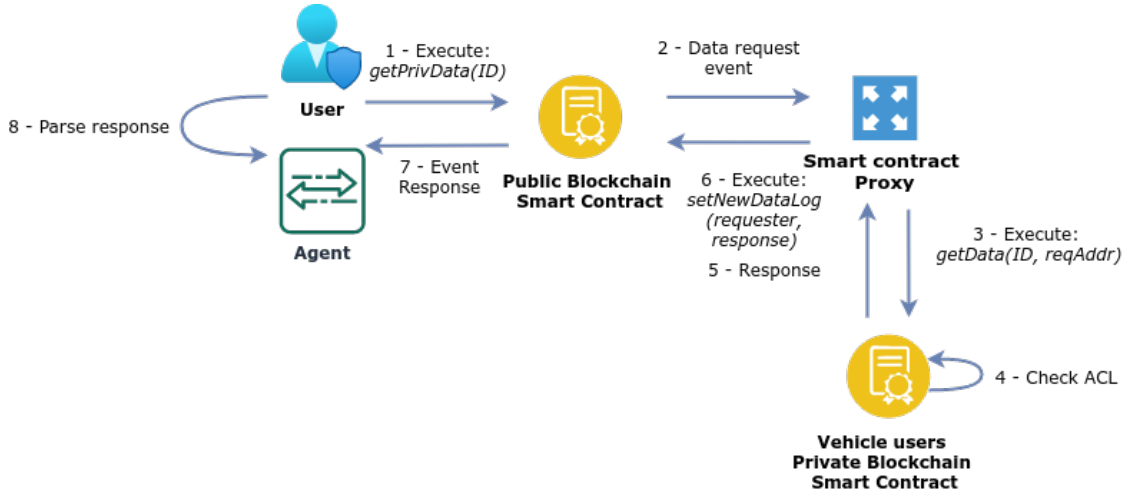

Figure 5.5: Data access request made by an authorised node of the public blockchain, with access control in the PBVU's smart contract.

Data management

In Figure 5.6, the vehicle user requests the MQTT broker to delete specific data that was previously stored in Storj. The user sends the command: delete "data ID", which triggers the MQTT to delete the data with the correspondent ID. In a scenario of data modification, the process is similar. Although the command is: modify "data ID". The serial stored in the smart contract does not need to be changed since the referenced object is deleted (serial is invalidated) or modified (serial remains the same).

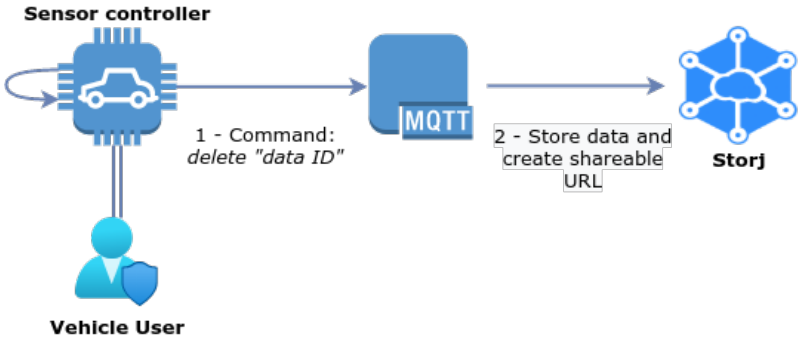

Figure 5.6: Data erasure request.

5.5 Conclusions

In this section, the privacy-enhancing proposal of this thesis was detailed. The solution consists of using MQTT as a more efficient messaging protocol to transfer IoT data generated by smart vehicles, which is stored in a decentralised storage solution and exchanged in a tiered architecture of blockchains. Blockchain, by design, provides security to data due to its decentralised and distributed architecture. However, to enhance the security and privacy of data, access control is performed using smart contracts. The data is encrypted using symmetric cryptography. The owners of the data (vehicle users) are responsible for defining the ACLs of their data.

The data is securely and privately stored in Storj that functions as the off-chain storage solution to prevent storing actual data in-chain due to the immutable nature of blockchains. Then, to exchange the data securely and privately, a serialised access string is created for each data entry and stored in-chain in the PBVU. Public blockchain nodes need to be authorised to have access to the data. They need to execute one of the smart contract's functions (*getPrivData()*) to create a data access request. When executed, it relays the request to the smart contract proxy. Additionally, when data is sent outside of the private blockchain, any information about the owner of it is not included (blockchain address) to preserve the privacy of the vehicle user. However, in certain data types, the user has to share information about him in the data itself (e.g. the user has been involved in a crash).

In sum, this proposal intends to preserve the user's privacy, location data, and other types of data generated by him through various technologies that complement each other.

Chapter 6

Experimental evaluation and analysis

This chapter will discuss the experimental evaluation of the proposed architecture based on the previously defined methodologies to infer if the defined requirements were met and formulate future improvements.

6.1 Evaluation strategy

Some validations will be performed to verify if the requirements were accomplished and if the system works as intended. Quantitative, observation and qualitative approaches will be used, based on the type of requirement. The quantitative results will be evaluated by comparing some of the results in scenarios with different test conditions. The observation methodology will be used for more functional and security related requirements to infer if they are implemented. Lastly, the qualitative evaluation will be analysed based on the other proposals, classical systems, and other scenarios where they are can be helpful, which will consist of identified possible limitations and vulnerabilities that these requirements can mitigate.

Finally, the proposal will be compared with the GDPR's required characteristics identified in [74], which will assess the GDPR requirements that were achieved, and future implementations to provide more user-centricity and privacy to the proposal.

6.2 Experimental scenario

The experimental scenario (Figure 6.1 consisted of using one physical machine hosting 4 VMs, one local area network with a stable 1Gbps Ethernet connection between the router and the host machine, and a 200Mbps uplink to the internet, to communicate with the public blockchain and Storj.

6.2.1 Technologies

As mentioned in subsection 5.4.1, this system will be implemented in a real scenario since simulators and emulators are limited in the number of available functionalities and development options. A real implementation scenario provides more flexibility to accommodate all the technologies and interactions of this architecture. The technologies and devices taken

Table 6.1: Functional requirements evaluation strategy.

Requirement	**Type of evaluation**	**Methodology**	**Objective**
Intermediation between blockchains	Observation	Log verification	Verify the logs created by the system components to assess if the smart contract proxy is executing cross-blockchain communications as designed.
Decentralised storage	Qualitative	Comparison with other storage systems (e.g., IPFS based systems)	Compare the implemented storage solution with other systems (e.g., centralised or decentralised storage) and the previously identified proposals.
Data management	Observation and Qualitative	Verify the well-functioning of the user's data requests. Comparison with other identified blockchain-based proposals.	Verify if data can be accessed, delete and modified by the owner. This requirement will be compared with other blockchain-based proposals previously identified and verify their level of GDPR compliance.
Access control	Observation	Log verification	Verify the logs created by the system components to assess the well functioning of this requirement in cases where unauthorised and authorised nodes try to request data.
Accountability	Observation	Smart contract request execution	Verify if the log events are being generated when new data is added to the system.
Flood prevention	Quantitative	Metric measurement	Calculate the amount of cryptocurrency used to perform access data requests to validate the flood prevention feature. The metric to be used will be the average fees per transaction (eth/transaction), which will assess the cost of running a data request flood attack (e.g., DoS) in the public blockchain.
Request processing	Observation	Log verification	Check the logs generated by the MQTT broker to verify if the requests are being served, and verify the correct implementation of these requests (e.g., if data is stored, modified or deleted correctly).

Table 6.2: Security requirements evaluation strategy.

Requirement	**Type of evaluation**	**Methodology**	**Objective**
User controlled privacy	Qualitative	Analysis and evaluation based on the comparison with other systems	Compare it with classic systems and the identified proposals. It has the purpose of assessing the quality and importance of this requirement since it is a step forward to have a more GDPR compliant system.
User anonymity	Qualitative	Analysis and evaluation based on the comparison with other systems	Compare it with classic systems that do not use blockchain technology and identify the vulnerabilities of not using this feature and how this requirement can mitigate those vulnerabilities.
Location and data privacy	Qualitative	Analysis and evaluation based on the comparison with other systems	Compare this requirement with classic systems and other proposals (e.g., centralised storage: databases or cloud storage).
Data confidentiality	Observation	Verify the communications	Verify if the data and communications between the sensor controller and the MQTT broker are encrypted, and if data is encrypted when stored and received by the authorised nodes.
Authentication	Observation	Log verification	Test connections to the MQTT broker with authorised and unauthorised nodes and check the logs to verify the well-functioning of this functionality.
Authorisation	Observation	Log verification	Test topic subscriptions and publishes in which the user is unauthorised, and check the logs to verify the well-functioning of this functionality.

Table 6.3: Non-functional requirements evaluation strategy.

Requirement	**Type of evaluation**	**Methodology**	**Objective**
Performance	Quantitative	Time overhead measurement	The time overhead metric will be measured in various system interactions to measure the system's performance. With this metric, we can infer the overhead that this system, especially the Ethereum blockchain, introduces to certain requests.
Blockchain size-reduction	Quantitative	Transaction size measurement	The transaction size of each request and event of this system will be measured to assess if the tiered architecture of blockchains allows reducing the size of the public blockchain, and if this reduction is significant. Thus, if the reduction is significative the blockchains' growth is not as pronounced with this architecture.
Bandwidth consumption reduction	Quantitative	Measurement of bandwidth consumption	The network traffic overhead allows measuring the traffic output generated by requests made by the sensor controller. It provides a way to verify if using MQTT as the middleware between the device and the blockchain saves bandwidth, which is important since some wireless mediums used by smart vehicles can provide low bandwidth. This experiment will be performed in two scenarios. In one scenario, the sensor controller uses the default method MQTT to publish data, and in the other, it uses a blockchain method of communication.
Interoperability	Observation	Log analysis	Through the logs generated by the MQTT broker and the sensor controller, it is possible to verify the communications between these devices.

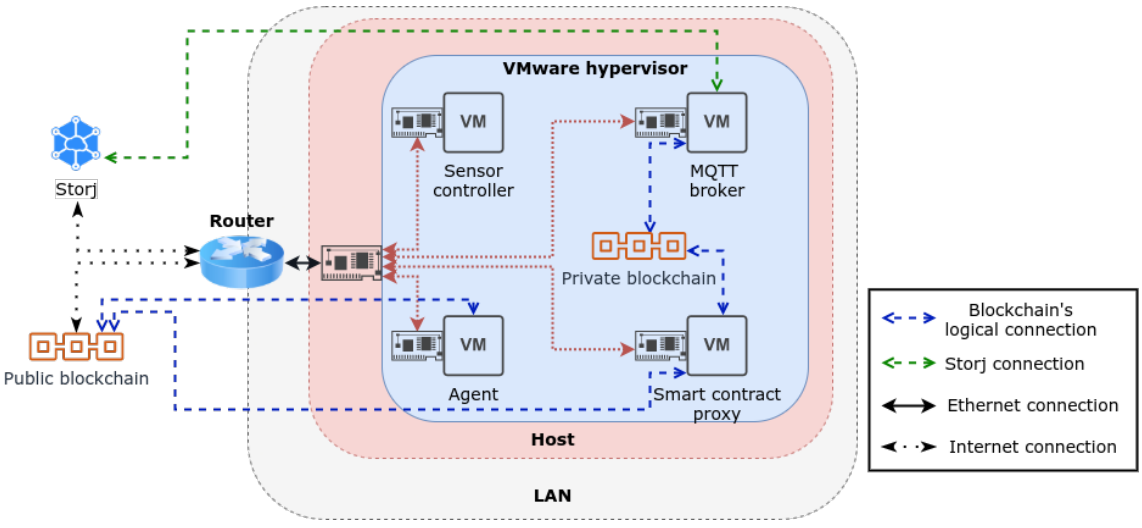

Figure 6.1: Topology of the experimental scenario.

into account to implement this system are Ethereum, MQTT, cryptography, a machine with plenty of computational resources, Virtual Machine (VM) technology and developed software integrated into the devices.

Ethereum

In terms of blockchain, Ethereum will be used as the blockchain technology in this system since it allows the deployment of smart contracts [43] and is considered a mature blockchain (has been used publicly since 2015). The public blockchain and the PBVU are Ethereum blockchains. The Ethereum smart contracts can be developed using various languages (Solidity, Vyper, and Yul [47]). However, the language that offers more functionalities and documentation is Solidity [42], which is the one going to be used. In this system, smart contracts will be developed using this language and will be deployed in the blockchain using Remix [23], which allows to compile and deploy smart contracts in a blockchain network.

The Ethereum private blockchain will be created using Go Ethereum (*geth*) [29], which implements the Ethereum protocol, and executes a private blockchain. Furthermore, *geth* allows implementing various types of nodes: full nodes, light nodes, and miner nodes. Each node of the system will use *geth*, and in the case of nodes participating in two Ethereum blockchains (e.g., smart contract proxy), they will run two instances of *geth*. The Ropsten Ethereum testnet will be used as the public blockchain since it is the testnet that is the most similar to the Ethereum mainnet, and the coin does not have a real monetary value. Therefore, it is free to make transactions and implement smart contracts.

Furthermore, Ethereum has a fee system that has a measurement unit denominated as gas [5]. This gas is charged to the blockchain user for each action that makes a state change, such as transactions, smart contract deployment or smart contract execution. This gas is collected and distributed to the miner nodes as a reward for their computational effort (PoW), which helps secure the network by preventing floods. The higher the computational requirements to do the operation, the higher is the cost. The gas cost is denoted as Gwei, which is equal to 0.000000001 ETH (10^{-9} ETH).

MQTT

The MQTT broker will be implemented using HiveMQ [30], because it is open-source, allows the deployment of plugins to extend its features and provides functionalities to improve its scalability (e.g. Broker clustering [88]). An example of a plugin is: Role-Based Authentication Control (RBAC) authentication plugin [31], which allows the authentication and authorisation of users. Furthermore, it allows using TLS to encrypt the communications. Therefore, it allows fulfilling the MQTT requirements defined.

Cryptography support

In order to provide confidentiality to the data, a symmetric cryptography algorithm will be used. The elected algorithm is Advanced Encryption Standard 256-bit key (AES-256) since it is considered a secure algorithm to encrypt data by certified entities, such as the National Security Agency (NSA) [35]. It was developed by Vincent Rijmen and Joan Daemen and was published in 2001 by the National Institute of Standards and Technology (NIST). The purpose of its development was to replace the Data Encryption Standard (DES) algorithm, which was considered insecure in the 1990s (the key was breakable in only 22 hours). In

contrast, it is estimated that an Advanced Encryption Standard (AES) key takes billions of years to be cracked.

AES is a block cypher algorithm, which uses 128-bit blocks. It encrypts data by performing consecutive permutations and substitutions of bits of the data and adds additional security by generating new keys in each modification stage. All the nodes need the same secret key to decrypt the data. As it is a symmetric encryption algorithm, it is faster and more efficient to execute.

The asymmetric cryptography algorithm that will be used to encrypt the symmetric key to exchange it securely is the Elliptic Curve Integrated Encryption Scheme (ECIES) framework with the secp256k1 algorithm [3]. Since the key pair of Ethereum accounts are created using the Elliptic Curve Digital Signature Algorithm (ECDSA) with the secp256k1 algorithm [2, 4], they can be leveraged to use ECIES. In this cryptographic scheme, a key pair is used, the private and public key. The data is encrypted using the public key and decrypted using the private key.

Virtualisation support

A machine with enough memory and storage capacity and processing power will be used to consolidate a set of VMs, which will represent each one of the architecture's devices. The VMs will be deployed in a type 2 hypervisor (installed on top of an OS), such as the VMWare workstation player. [37].

Software

The system's application is composed of various pieces of software running on the various nodes of the architecture. Each type of node has its application requirements. Hence, they use and interact with different technologies. However, some common technologies are going to be used, such as *Web3.py* [8].

To represent an end-user application, NodeJS [12] will be used to create a web application. This application will be deployed in the sensor controller. It will provide an User Interface (UI) where the user can submit the ACL for each data type that he desires, and perform data management requests, such as access, modify and delete his data. It communicates with the MQTT broker to process and send the user's requests.

The request handling in the smart contract proxy and MQTT broker is performed by a Python [22] application. It will use the *Web3.py* library to interact with the *geth* node/s in execution in the device, either to perform smart contract calls or to await for events. The agent node will also have a similar application to make data access requests and wait for blockchain events.

As mentioned before, Solidity will be used to develop the smart contract.

6.2.2 Experimental scenario components

The host machine has an i7-7700HQ CPU 2.80GHz with eight logical cores and 16 GB of ram. Since this machine does not provide optimal resources to do performance tests, the VMs were not all turned on at the same time to maximise the resource distribution. The OS of the VMs was Ubuntu 16.04 without a graphical interface to minimise resource

consumption. The experimental architecture is composed of:

- **Private blockchain:** This blockchain was created and executed locally using *geth* on the MQTT broker and the Smart contract proxy. The consensus protocol selected was the PoA since it requires less computational resources to mine blocks. Furthermore, it was configured to mine new blocks every 3 seconds, and the rest of the configurations were default.
- **Public blockchain:** To achieve more realistic results, the public blockchain was not implemented locally (via simulation of the Ethereum mainnet with *geth* nodes). Instead, the *Ropsten* Ethereum testnet was used. It was selected because it is the testnet that is the most similar to the Ethereum mainnet since it uses the same consensus protocol (PoW) and configurations. Moreover, as it is a testnet, its coin is used for test purposes. Hence, the transactions and interactions with smart contracts do not have a monetary cost. On the other hand, the mainnet (Ethereum) was not considered because it requires real ether, which could become very costly to implement this system and perform tests.
- **Sensor controller:** This device uses a web application that was developed in Node.js to communicate with the MQTT broker and process his requests. In this application, the user can submit the ACL for each data type that he desires, and perform data management requests, such as access, modify or delete his data. An "add data" option was added to the application to simulate the behaviour of a smart vehicle. It was used to test the data publish requests into the system and for load testing. A direct blockchain communication option was also implemented in this application, using the *web3js* API in conjunction with a *geth* instance. This alternative communication method aims to infer the differences between using a MQTT middleware to communicate with the blockchain or direct blockchain communication in a constrained environment. This VM represents a constrained device. Thus, only one CPU core and 2 GB of ram were allocated.

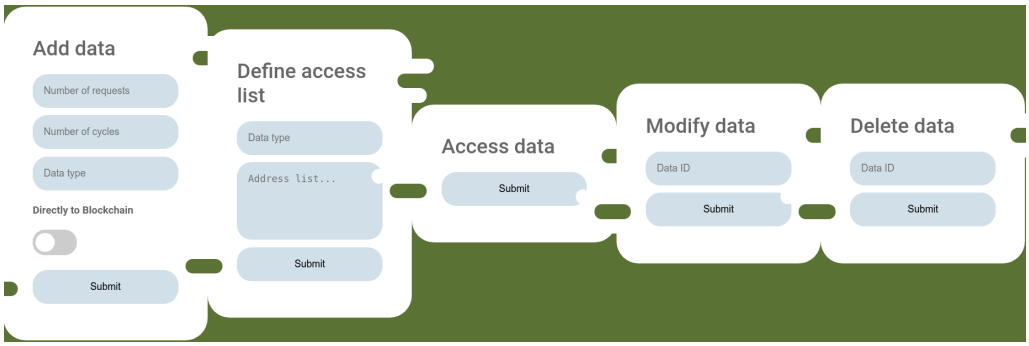

Figure 6.2: Web application interface.

- **Access network emulation**, a cellular network emulator was used [73] to provide a more realistic environment. It consisted of using a NetEm (Network Emulator) [11] profile to emulate a 4G network with roaming [9]. This profile imposed a delay distribution on the network interface, which emulates the connectivity latency of these cellular networks.
- **MQTT broker:** This VM is responsible for handling the sensor controller requests by receiving them via the MQTT protocol. Thus, it is running a MQTT broker server, HiveMQ, and a python script with the *web3* API to use the *geth* instance

to interact with the private blockchain. The python script, also used the *libuplink-python* library [20] to communicate with the Storj storage system. Since it only works at the private blockchain level it only runs one *geth* node instance. In terms of resources, four CPU cores and 3 GB of ram were allocated.

- **Smart contract proxy:** As mentioned previously, it is the middleware between the private blockchain and public blockchain. Hence, it is connected to both of them through 2 *geth* instances. This node is also running a python script in conjunction with the Request processing API, which allows interaction with *geth* nodes to handle the requests from both blockchains simultaneously (listen to events and execute smart contract functions). This VM does the most resource-intensive tasks. Therefore, four CPU cores and 4 GB of ram were allocated.

- **Agent:** This VM works at the public blockchain layer. Thus, it runs only 1 *geth* node instance. Similarly to the other two previous nodes, it also runs a python script with the Request processing API to listen to events and makes data requests to smart contracts. The resources allocated were four CPU cores and 3 GB of ram.

6.3 Test conditions

The tests were executed with a fixed duration time of 300 seconds and with various requests rates (1, 10 and 50 seconds per second) to test the system's performance in various load scenarios. Each of these test conditions was executed five times to get a bigger result sample. Thus, reducing the experimental error. The requests used for the performance tests were the *publish data*, in the sensor controller, and the *get data*, in the agent node. These requests resemble the submission of data by the smart vehicles (*publish data*) and the request of data (*get data*) by public nodes (agents and users), which represent the majority of requests that will be performed in this system. Therefore, they are the most relevant ones to conduct performance tests.

In the sensor controller, two communication methods were used: MQTT and direct blockchain communication. The NeTem profile used emulated 4G connectivity with a good quality signal and roaming to simulate a moving device communicating with different antennas. In [9] more profiles are present, such as profiles that emulate 3G connections and other signal quality levels. Although, 4G is the most adopted cellular network technology.

6.4 Metrics

A set of metrics were obtained from each architecture node to analyse this system's outcomes. These metrics are identified in the following table.

Table 6.4: Metrics measurement strategy.

Requirement	**Metric**	**Measuring unit**	**Measuring tool**	**Strategy**
Flood prevention	Transaction cost	Gwei per transaction	*Web3.py* Python library [8]	*Web3.py* library [8] offers an API that allows obtaining information related to transactions created by the *geth* node, which was used to obtain the transaction costs of data requests generated by the public blockchain nodes. The cost values were obtained in Gwei since the base currency unit of Ethereum, ether (ETH), is too large to be used to represent the costs of each transaction. Gwei denotes a ninth power of the fractional ETH. This calculation will be implemented in the software application of the agent node.
Performance	Time overhead	Milliseconds (ms)	Time Python library	The *time* library offers methods that can obtain the current time in milliseconds (e.g, perf_counter() and time()). These methods were integrated into the devices' applications to measure certain request's completion time.
Blockchain size-reduction	Transaction size	Kilobyte (kB)	*Web3.py* Python library [8]	The size of each state-changing transaction was measured using the Web3 API.
Bandwidth consumption reduction	Bandwidth consumption	Kilobits per second (kbps)	Nethogs Linux tool [10]	Nethogs [10] is a Linux tool that allows tracking the amount of bandwidth that is being used per process in the OS. It will be implemented in the sensor controller application to quantify the bandwidth consumption of *data publish* requests. The metric is going to be measured in two scenarios: a scenario where the sensor controller communicates directly with the blockchain, using a *geth* node, and in another one, it will communicate using the default procedure of this system: MQTT.
-	CPU and memory usage	Usage in percentage (%)	psutil python library [17]	The CPU and memory usage were measured using a python library named psutil [17] which measured the resource consumption in each of the system's nodes when an experiment was running or when a certain task was ongoing.

6.5 Analysis of results

In this section, the results obtained in the experiments are going to be analysed. Firstly, the *publish data* request results obtained in the sensor controller and MQTT broker will be used to compare the two communication methods used in the sensor controller (MQTT and blockchain). This result will help to infer if using MQTT is more beneficial for resource-constrained environments. The remaining results will be examined to evaluate the system's performance, validate certain requirements and formulate possible improvements to this system.

6.5.1 Comparison between MQTT and blockchain

Table 6.5 and Figure 6.3 display the results regarding the sensor controller tests performed using the two communication scenarios. By analysing the results, it is clear that the blockchain communication method is more resource-intensive, especially memory usage.

Therefore, MQTT is a more lightweight approach for this system because the broker handles the sensor controller requests. Thus, some computational effort is taken away from it.

Table 6.5: Resource consumption results in the sensor controller using two communication methods.

Rate	Communication method	Bandwidth consumption (Kbps) / standard deviation	Mean CPU usage (%) / standard deviation	Mean Memory Usage (%) / standard deviation
1 req/s	MQTT	0,667 / 0,148	5,367 / 1,629	14,456 / 0,063
	Blockchain	4,073 / 1,965	10,252 / 7,268	69,191 / 2,440
10 req/s	MQTT	7,580 / 7,813	8,233 / 4,887	15,048 / 0,435
	Blockchain	14,744 / 8,654	30,689 / 21,507	96,098 / 3,360
50 req/s	MQTT	13,217 / 10,198	12,853 / 6,120	15,817 / 0,444
	Blockchain	64,100 / 28,767	29,997 / 20,302	98,261 / 2,032

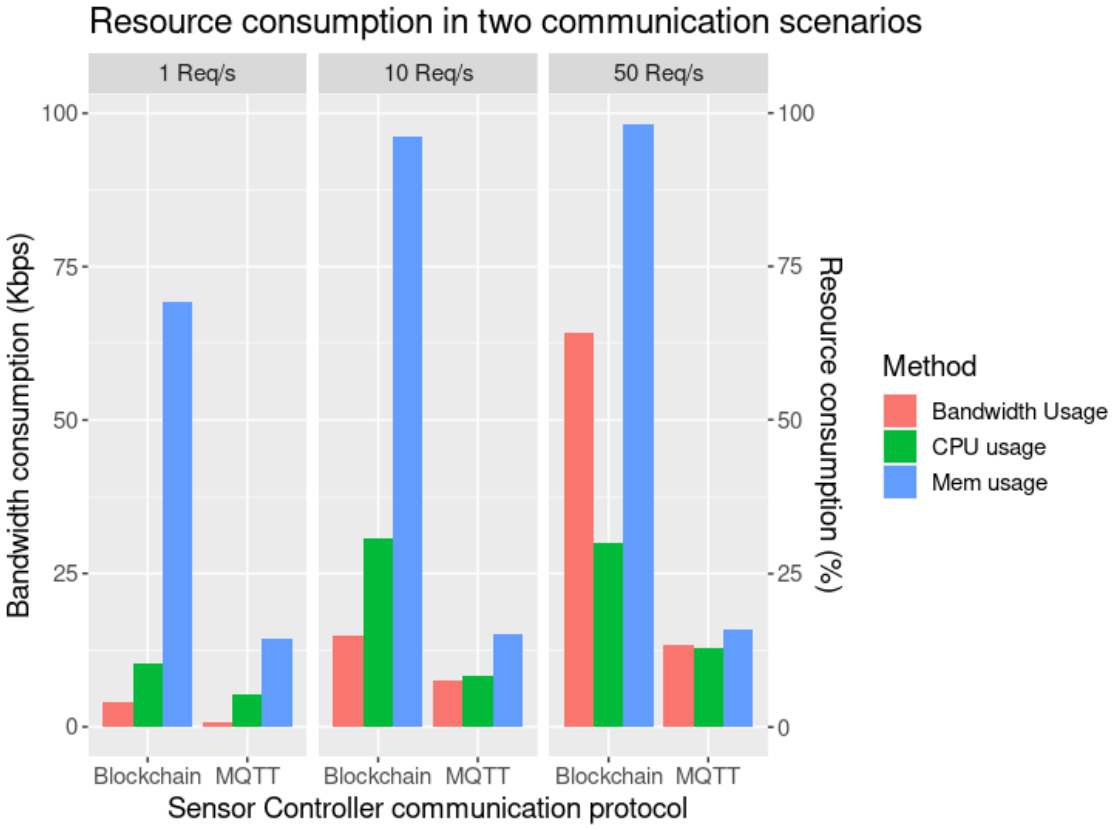

Figure 6.3: Graphic with the resource consumption results of the sensor controller.

Although the mean results seem to demonstrate that the MQTT is a more lightweight methodology to communicate in this system, statistical testing was done to compare the variances of each group of results and to provide a more statistical foundation to our conclusions.

Firstly, the results are tested to check if they comply with the parametric testing assumptions: homogeneity of variance (all comparison groups have the same variance) and normal

distribution (normality). The tests used to verify the assumptions were: Shapiro-Wilk to test normality and Barlett to test [80] the homogeneity of variance. By testing the samples, we can infer that neither of the assumptions is satisfied. Therefore, a non-parametric hypothesis testing was applied. The chosen statistical test was a non-parametric two-way ANOVA alternative, which was the randomisation test with unrestricted permutations [51]. It allows statistical analysis using two-way ANOVA in samples that do not follow a normal distribution. In addition, it analyses samples with various independent variables (communication method and the request rate) and how they affect the dependent variable (bandwidth consumption, CPU usage and memory usage). It also tests three null hypotheses simultaneously. In our case, they are:

- No difference in means due to the communication method.
- No difference in means due to the request rate.
- No interaction of factors.

The statistical test was performed for each dependent variable (bandwidth consumption, CPU usage and memory usage) with a significance level of 5%. As expected, all the null hypotheses were rejected. The p-value is less than 5% in all hypotheses. Thus, the assumption that the MQTT communication method is more lightweight than the blockchain method is sustained.

Despite MQTT requiring less computational power, it adds more time overhead to *publish data* requests, and the results have more variability as the frequency of the requests grow (Figure 6.4). This behaviour occurs due to the extra layer of latency and queuing that MQTT imposes, making the time overhead results more bursty.

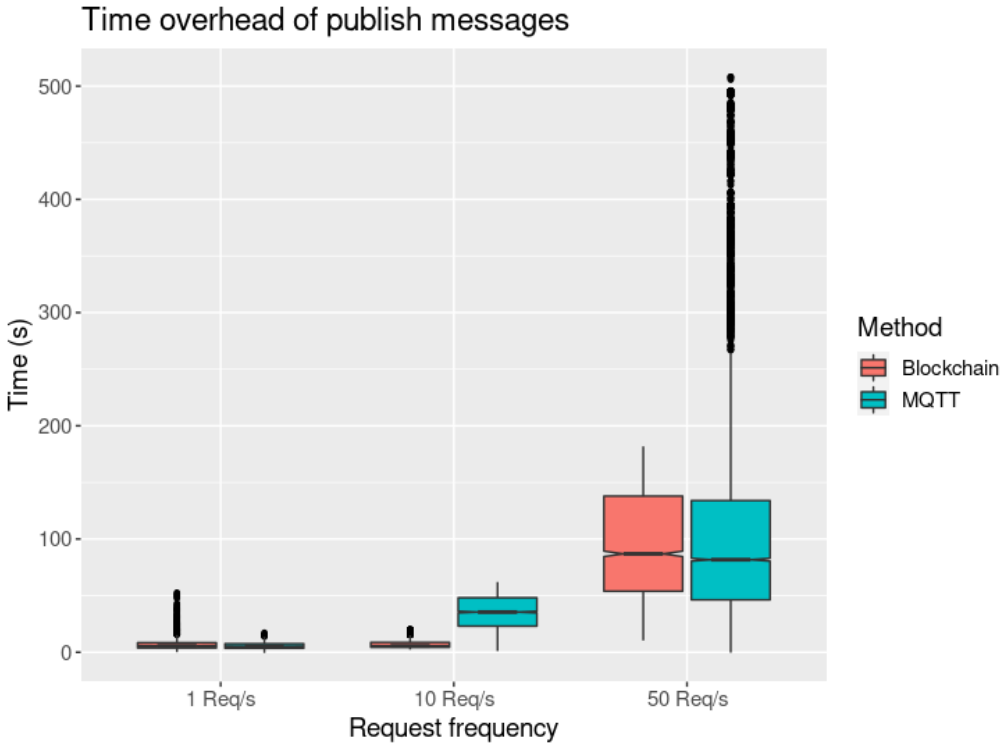

Figure 6.4: Time overhead results of the sensor controller.

6.5.2 Performance evaluation

Figure 6.5 and 6.6 illustrate the results of the data request load tests performed by a public node. Since these requests are performed in the public blockchain, the time overhead of each request depends on its performance. As the request frequency increases, the time overhead becomes more unstable and gets higher, as displayed in Figure 6.5. The CPU and memory utilisation of the smart contract proxy follows the same pattern as the time overhead of the requests, bigger values and more variable values. At 50 requests per second, the smart contract proxy goes beyond its throughput limit (in this scenario conditions) since the mean response time becomes very high, as well as the resource consumption variance. This behaviour occurs because many concurrent threads are in execution in the system and because they are released and created with an irregular rhythm, the CPU usage variability increases.

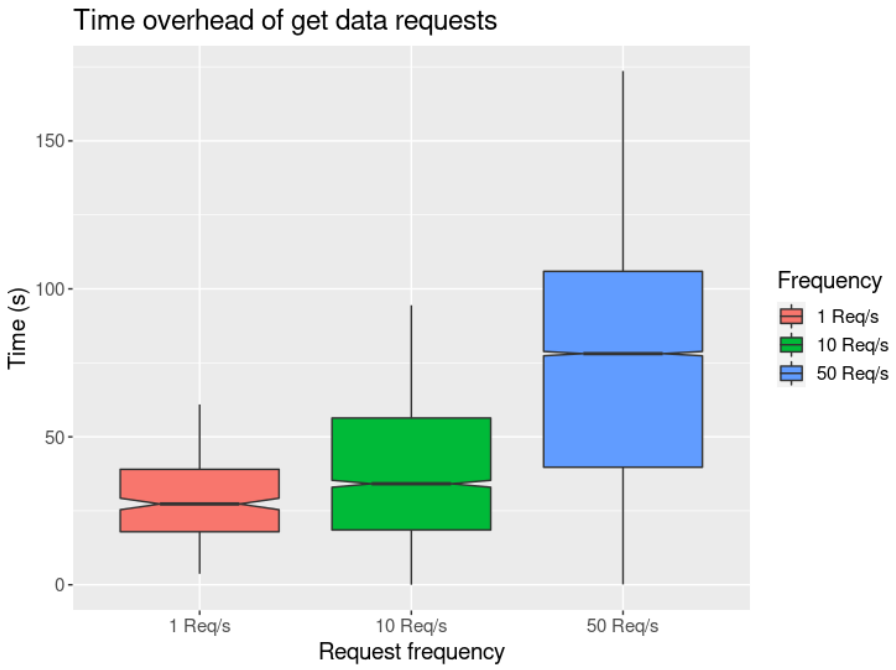

Figure 6.5: Time overhead results of the agent.

By comparing the results between Figure 6.4and 6.5, the public blockchain proves to add more latency to the requests than the private blockchain. The mean results with 1 request per second in the private blockchain are 5,689s (using MQTT) and 6,997s (using direct blockchain communication), and in the public blockchain is 35,085s. Despite the fact that the *get data* request performed in the public blockchain requires two transactions to be completely handled, the mean of each transaction is roughly 17,543s, which is approximately two times higher than in the private blockchain. This behaviour is justified because of the consensus protocol and by the dimension of the blockchain itself.

The private blockchain uses the PoA, which is more scalable since the miners are considered trustworthy. Thus, it requires less computational effort to mine blocks. Hence, it allows more transactions to be mined per second. It also does not have concurrent transactions, in this experimental scenario, from other nodes to be mined, which results in less congestion. On the other hand, the public blockchain (*Ropsten* testnet) uses the same consensus as the Ethereum mainnet, which is the PoW. This protocol is more resource-intensive and

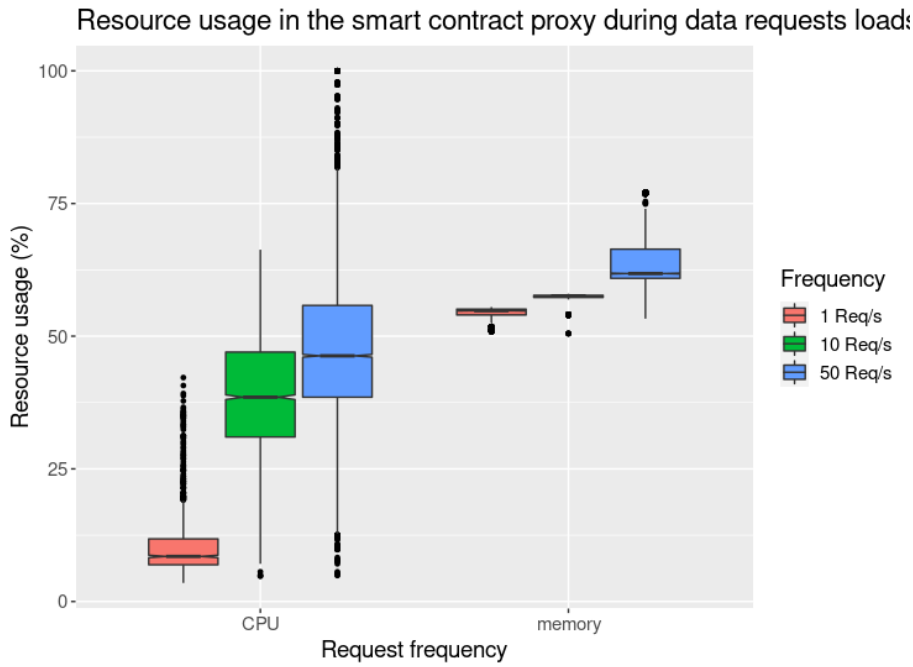

Figure 6.6: Resource consumption results of the smart contract proxy.

does not rely on trust. Therefore, to have more security and decentralisation, it provides a smaller throughput. Moreover, this testnet does not have the saturation level of Ethereum, but the concurrent transactions created by other nodes can also interfere with the latency since the transactions can be queued.

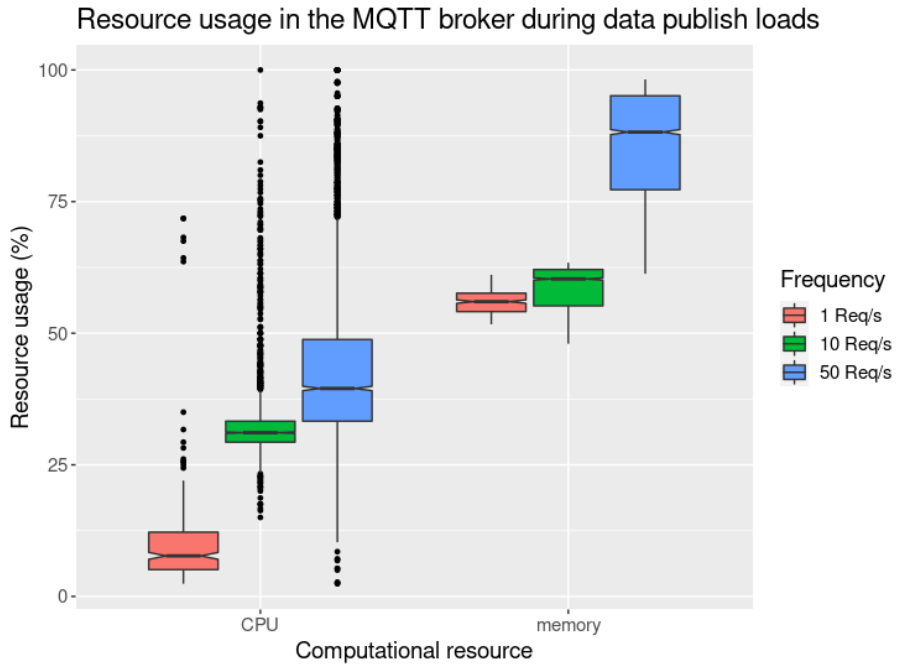

Figure 6.7: Resource consumption results of MQTT.

The MQTT broker computational resources' utilisation is displayed in Figure 6.7. Similarly

to the previous results, the node utilises more resources as the request rate increases. The same is observable in Figure 6.4 in its time overhead results, which follows a similar pattern as the smart contract proxy (Figure 6.6).

6.5.3 Blockchain size reduction evaluation

In Table 6.6 is registered the transaction size of the requests and events that occur in the blockchain system of this proposal. These requests and events are distributed in both blockchains. The private blockchain produces the biggest transactions since this blockchain is responsible for handling the vehicle users' data (data produced by vehicles, ACLs, keys and other requests). It also is responsible for storing information in the blockchain and querying the smart contract proxy to advertise information in the public blockchain. Therefore, if only one blockchain is used, its size would increase significantly as the number of nodes of the system grows. It increments the blockchain size and the throughput needs, resulting in more hardware to run this system. Hence, using a more distributed architecture of multiple blockchains is more beneficial for the system's scalability in the long run compared to using a single blockchain.

Table 6.6: Mean transaction size of the various requests and events in Kb.

Private Blockchain			**Public blockchain**			
addData mean and sd of tx size (Kb)	defineACL	exchangeKey	createLog	createTransaction	dataRequest	dataResponse
90,924 / 1,690	1,459 / 0,000	7,557 / 0,048	0,138 / 0,000	0,656 / 0,001	0,275 / 0,000	0,315 / 0,000

6.5.4 Flood prevention

The cost of each data request transaction is mostly fixed. Although it can have a slight gas fees variability, it is not expressive enough. The cost in Gwei is 730,000039681, which converted to ether is 0,000000730000039681 (around 0,0013€ at the time of writing – 20/04/2020). It seems like a small fee amount. However, it can become costly to run brute force attacks or DoS attacks since thousands or millions of requests are performed. Moreover, as the congestion of the network increases, the fee prices also increase.

6.6 Requirements validation

The experiments allowed to obtain statistical results related to performance and other metrics. However, they were also used to validate functional, and privacy and security related requirements. In the appendix A, screenshots of logs are present to provide visual feedback of the well functioning of some of the defined requirements.

Table 6.7: Functional requirements validation.

Requirement	Methodology	Interpretation of results
Intermediation between blockchains	Log verification	By verifying the logs generated in the various cross-chain requests and responses of the system, the well functioning of this requirement is validated.
Decentralised storage	Comparison with other storage systems (e.g., IPFS based systems)	As mentioned before, the implemented decentralised storage solution was Storj. This solution provides enhanced security features by default (e.g., encryption, data loss prevention and access control), making it more secure than centralised storage systems. Considering other decentralised storage systems, such as IPFS [49], it is more GDPR compliant since data can be effectively deleted. Using the blockchain as storage [95] is even less GDPR compliant since it is a transparent and tamper resilient system. Therefore, between all the previously mentioned approaches, this one seems to be the best at safeguarding the privacy and security of the user's data.
Data management	Verify the well-functioning of the user's data requests. Comparison with other identified blockchain-based proposals.	The logs produced by these requests demonstrate the correct implementation of this requirement, enhancing the level of GDPR compliance of this system.
Access control	Log verification	By performing authorised and unauthorised requests, it is observable that the access control mechanism functioning in the smart contract is observable through the logs.
Accountability	Smart contract request execution	The events containing the log information of newly added data are observable and parsed by public nodes. Thus, the requirement is validated.
Flood prevention	Metric measurement	In section 6.5.4, the gas costs results of *data requests* were discussed, and it is possible to affirm that this functionality is working and can be beneficial to prevent high volumes of requests.
Request processing	Log verification	By checking the logs, we can infer that the MQTT broker correctly handles the sensor controller requests.

Table 6.8: Non-functional requirements validation.

Requirement	Methodology	Interpretation of results
Performance	Time overhead measurement	The results associated with the time overhead of the *get data* and *publish data* proved that using a private blockchain to handle requests generated by the sensor controller offer less latency.
Blockchain size-reduction	Transaction size measurement	Through the obtained results, we can infer that using a more distributed system of blockchains is more beneficial since the sizes of the blockchains do not grow as much as using only one blockchain, and they are less congested.
Bandwidth consumption reduction	Measurement of bandwidth consumption	The statistical analysis concluded that MQTT communications use less bandwidth than the blockchain protocol to interact with the system. However, it increases the latency of the requests.
Interoperability	Log analysis	This system provides MQTT communications, which are supported by many devices, to interact and use this system. Therefore, this system is interoperable with devices that do not support blockchain technologies. Future, different device types can be used for more in-depth and realistic tests (e.g., Raspberry pi).

Table 6.9: Security requirements validation.

Requirement	Methodology	Interpretation of results
User controlled privacy	Analysis and evaluation based on the comparison with other systems	One of the GDPR policies dictates that the user should be able to specify his privacy preferences over his data. However, in most of the classical applications and some of the proposals identified ([49, 62, 76, 95]), the users cannot define their privacy preferences, which could result in data being used by other entities without their consent. In this system, the user has the freedom to specify the list of public nodes that can access their data, which enhances his data privacy.
User anonymity	Analysis and evaluation based on the comparison with other systems	Usually, applications gather information about the user for Know Your Customer (KYC) purposes or other goals. Even if this information is being used for legitimate purposes and has the user's consent, it is still prone to attacks. Therefore, implementing enhanced anonymity by design could mitigate these privacy threats, which makes the anonymous nature of blockchains beneficial to improve the overall privacy of users in this system. It is only possible to know to whom the address belongs to if the owner advertises it.
Location and data privacy	Analysis and evaluation based on the comparison with other systems	The proposal [95] stores data in the public blockchain, which imposes a threat to the privacy of the data since it is publicly available to the public nodes. Other solutions use other storage systems that do not guarantee the safety of the data (e.g., centralised storage systems). In this system, data is stored outside the blockchain in a decentralised system, making it more private and secure. Inherently from the user anonymity, the location information and data generated by smart vehicles become anonymous. Furthermore, when data is being logged in the public blockchain, the blockchain address of the data owner is not disclosed as well, which prevents profiling and tracking attacks. Although, in some sporadic events, the user's sensitive information needs to be included in the data, e.g., accidents in which he was involved. There should be a consensus between these certified authorities and the users in these cases, but it is out of the scope of this proposal.
Data confidentiality	Verify the communications	This requirement is implemented as intended since the data is encrypted when in transit or stored in Storj and can only be decrypted with the specific symmetric key.
Authentication	Log verification	The tests performed to the authentication system of MQTT proved to be working since users not included in the database cannot authenticate.
Authorisation	Log verification	Various roles were created to restrict the operations of each user. For example, a user can only publish or subscribe to the topic that corresponds to its blockchain address.

6.7 Privacy analysis

Table 6.10 provides an overall comparison between the previously analysed blockchain-based proposals and ours. The characteristics used to classify are from the privacy framework identified in [81] and from the GDPR survey [74].

By looking at the framework characteristics of Table 6.10, it is noticeable that our proposal still lacks some privacy aspects. The proposal [63] manages to cover more aspects of privacy. Nevertheless, it is safe to say that ours is more user-centred than the others since we offer more data controllability to the user. For example, *CR7* dictates that the user must be able to erase or modify his data, which was implemented. Hence, our solution can be considered to be more GDPR-compliant. However, it still needs further development, e.g., provide more transparency to the user over his data, how it is being used, and implement a method to establish agreements between the user and the entities.

Table 6.10: Comparison of the classification of the blockchain related privacy proposals identified with our proposal.

Proposal	Openness, transparency, and a specified purpose	Identity privacy	Temporal and location privacy	Query privacy	Access control	Interoperability	Data minimisation	Accountability	Security of data	GDPR [74]
[56]		✓		✓		✓			✓	CR1 CR11
[63]	✓	✓	✓	✓	✓	✓	✓		✓	CR1 CR2 CR4 CR11
[76]		✓		✓		✓			✓	CR1 CR11
[95]	✓	✓	✓	✓		✓			✓	CR1 CR2 CR11
[62]	✓	✓		✓	✓		✓		✓	CR1 CR11
[49]		✓	✓	✓		✓			✓	CR1 CR2 CR11
Our proposal		✓	✓	✓	✓	✓			✓	CR1 CR2 CR4 CR7 C10 CR11

6.8 Conclusions

These experiments allowed us to evaluate the system's current state, test and infer other approaches, and formulate future improvements.

The current architecture offers high latency to certain requests, which is unsuitable for applications with low latency requirements. However, as explained before, it is affected by the underlying blockchain technology. Hence, the throughput and latency of this system are bounded by its limits. Moreover, blockchain is still a technology in development. In analogy with the internet, it did not provide a stable and high throughput at the embryonic stages of the internet as nowadays, it kept evolving with time. The same applies to blockchain. Ethereum has announced that it will launch an upgrade to its protocol called Ethereum 2.0 [48], which claims to enhance its throughput up to 100000 TPS [18]. Currently, there exist other blockchains that offer more throughput. However, they are less mature, such as Solana [94], which foresees up to 700000 TPS with a gigabit network connection [7].

Furthermore, the experiments were not performed in the most favourable scenario since more nodes and more computational resources are needed to perform a more in-depth analysis of this system's performance. Thus, the limitations of our experimental setup might have influenced the results. In addition, as concluded in [83], the performance of Ethereum private blockchains augments as computing resources increase. The software that handles the requests can also benefit from the increase of resources since it is multi-threaded.

The system manages to deliver high privacy and security to the users and their data by merging the benefits of various technologies. In this system, blockchain enhances, by design, the privacy of the user's identity and offers other security mechanisms that provide resilience to failures and data disruptions. The cryptography used in blockchains to generate addresses allows the creation of public keys and then use them to exchange private information, in this case, an asymmetric key to encrypt data, using asymmetric cryptography. MQTT also allows the implementation of privacy and security measures by

implementing authentication and authorisation and offers a more lightweight and interoperable communication approach for resource-constrained devices. Lastly, Storj enhances the privacy and security of the users' data and enables data management procedures to enhance GDPR compliance.

In sum, it is noticeable that exists a compromise between privacy and performance, mostly because blockchain is still a technology in development. Thus, it is not at the performance level of matured solutions (e.g., databases). Since the main focus of this proposal is to propose a privacy-preserving solution for IoT applications, this system manages to deliver the defined requirements. However, this compromise is acceptable since the privacy and security gains outweigh the performance.

Chapter 7

Conclusions and future work

In this thesis, we presented a state-of-the-art based on privacy-enhancing proposals oriented to IoT applications, in which they are evaluated, and some limitations are identified. This study was the basis for the formulation of our proposal since it builds on these proposals, while addressing critical security and privacy-related requirements.

Our proposal offers privacy to its users and data, and resilience to certain attacks and failures. It focuses on crowdsensing applications with mobility, such as smart vehicles, wherein the location information is generated to provide geospatial data to a system. Hence, these applications are prone to tracking attacks, in which the user movement can be traced and mapped to infer sensitive information (e.g., home or work location). Therefore, this proposal's core technology is blockchain since it allows to enhance the anonymity of users, by default, and the resilience to system's failures. Moreover, other technologies and mechanisms were integrated to enhance this system's privacy further and mitigate some of blockchain's limitations. This symbiosis between technologies also offers user-controlled privacy, in which the user can control the end nodes that can access his data and the exposure level of different data types. The attacks and vulnerabilities that this system mitigates are the following:

- **User profiling**. The user cannot be profiled since the blockchain address used inside the system to identify him is not linked to sensitive information or needs any of his information to be generated. In this experimental implementation, only a password was needed to generate the users' wallets using *geth*. Additionally, even in a scenario where the user's identity is disclosed, the owner's blockchain address or other information is not advertised when the data's serial is exchanged in the public blockchain. Therefore, it is highly unlikely that the data exchanged in this system could be used for user profiling.
- **User tracking**. The previously identified defence mechanisms also apply to this potential attack on the user's privacy. Thus, the data that could represent the user's movements or routines are private.
- **Linking attack**. An attacker cannot infer the identity of a user by analysing the previously exchanged data since the user's identity is preserved in this proposal.

The system offers protection from the previous attacks unless the data content has descriptive information that could reveal sensitive information about the user. However, it is out of the scope of this proposal since this information should be concealed at the source.

The experimental outcomes show that this architecture lacks performance due to bottlenecks related to the environment where it was implemented, blockchain technology, and developed software. Therefore, further studying and development need to be developed to enhance its performance and robustness.

The proposed and implemented work is a foundation for various IoT and non-IoT applications. Hence, more privacy, security and functional features can be developed to fortify this proposal. Therefore, we compiled a set of possible future improvements and implementations:

- Design a privacy-preserving solution that allows the public nodes to identify the data they can access without disclosing the ACL of the data entry or other private information.
- Implementation of a reputation system in which the misbehaved nodes (with the lowest reputation score) have to pay a pricier fee to request data, and vice versa. It could also be used to alert of possible risks when authorising nodes with a bad reputation, which could improve the GDPR compliance of the system (Section 3.2): CR9-Provide balance of privacy between the user and third parties; CR12-Provide ability to users to make informed consent choices; CR14-Communicate risks of data collection/inference to users.
- Use the transparent nature of blockchains to keep track of the nodes that request data access and inform the data owners by producing logs (CR16 – Provide transparency).
- System to manage the public keys of the public nodes, which could identify the nodes address (and other relevant information and their reputation. It could be implemented in a smart contract to provide integrity.
- Implementation of high availability mechanisms, such as load balancing between brokers, proxies and PBVU. Implementing this system in the cloud could also provide more availability and performance due to the horizontal elasticity of these environments.
- Explore the implementation in other blockchains (e.g., Cardano [38] and Solana [94]) that offer more throughput and other beneficial functionalities for this system.
- Perform a more in-depth study of the system's performance and scalability in a non-resource restrained environment, such as in the cloud.
- Furthermore, the system's current state could be optimised, specifically the developed software and the configuration of the blockchains and nodes.

In sum, despite our solution not being a complete system, it manages to use various technologies to offer a robust privacy-preserving architecture, not only for automotive applications but also for other IoT applications that require user-centricity and privacy.

References

[1] Blockchain service | microsoft azure. https://azure.microsoft.com/en-gb/services/blockchain-service/. (Accessed on 06/01/2021).

[2] Ecdsa: Elliptic curve signatures - practical cryptography for developers. https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages. (Accessed on 05/25/2021).

[3] Ecies hybrid encryption scheme - practical cryptography for developers. https://cryptobook.nakov.com/asymmetric-key-ciphers/ecies-public-key-encryption. (Accessed on 05/25/2021).

[4] Ethereum accounts | ethereum.org. https://ethereum.org/en/developers/docs/accounts/. (Accessed on 05/25/2021).

[5] Gas and fees | ethereum.org. https://ethereum.org/en/developers/docs/gas/. (Accessed on 05/25/2021).

[6] How to generate public and private keys for the blockchain | by artiom baloian | medium. https://baloian.medium.com/how-to-generate-public-and-private-keys-for-the-blockchain-db6d057432fb. (Accessed on 12/26/2020).

[7] Introduction | solana docs. https://docs.solana.com/introduction. (Accessed on 05/25/2021).

[8] Introduction — web3.py 5.19.0 documentation. https://web3py.readthedocs.io/en/stable/. (Accessed on 05/19/2021).

[9] marty90/mobile-latency-emulator: Emulate mobile network latency based on large scale real-world measurements. https://github.com/marty90/mobile-latency-emulator. (Accessed on 05/19/2021).

[10] nethogs(8) - linux man page. https://linux.die.net/man/8/nethogs. (Accessed on 05/24/2021).

[11] networking:netem [wiki]. https://wiki.linuxfoundation.org/networking/netem. (Accessed on 05/20/2021).

[12] Node.js. https://nodejs.org/en/. (Accessed on 05/19/2021).

[13] Open source blockchain for currencies & payments - stellar. https://www.stellar.org/learn/intro-to-stellar. (Accessed on 06/09/2021).

[14] Private networks | go ethereum. https://geth.ethereum.org/docs/interface/private-network. (Accessed on 06/01/2021).

[15] Proof of authority explained | binance academy. https://academy.binance.com/en/articles/proof-of-authority-explained. (Accessed on 06/01/2021).

[16] Proof of stake explained | binance academy. https://academy.binance.com/en/articles/proof-of-stake-explained. (Accessed on 06/01/2021).

[17] psutil · pypi. https://pypi.org/project/psutil/. (Accessed on 05/24/2021).

[18] Shard chains | ethereum.org. https://ethereum.org/en/eth2/shard-chains/. (Accessed on 05/25/2021).

[19] Slashing — bison trails. https://bisontrails.co/slashing/. (Accessed on 06/01/2021).

[20] storj-thirdparty/uplink-python: Python bindings for libuplink. https://github.com/storj-thirdparty/uplink-python. (Accessed on 05/20/2021).

[21] Teach, learn, and make with raspberry pi. https://www.raspberrypi.org/. (Accessed on 06/09/2021).

[22] Welcome to python.org. https://www.python.org/. (Accessed on 05/19/2021).

[23] Welcome to remix's documentation! — remix - ethereum ide 1 documentation. https://remix-ide.readthedocs.io/en/latest/. (Accessed on 05/25/2021).

[24] What is proof of work (pow)? | binance academy. https://academy.binance.com/en/articles/proof-of-work-explained. (Accessed on 06/01/2021).

[25] Can I delete my content from the network? - Help - discuss.ipfs.io. https://discuss.ipfs.io/t/can-i-delete-my-content-from-the-network/301, May 2017. [Online; accessed 12. Feb. 2021].

[26] General Data Protection Regulation (GDPR) – Official Legal Text. https://gdpr-info.eu, Sep 2019. [Online; accessed 11. Feb. 2021].

[27] area2invest | Decentralised Finance - the End of the Traditional Financial Economy? https://www.area2invest.com/decentralised-finance, May 2020. [Online; accessed 9. Jan. 2021].

[28] Bitcoin blockchain size chart — Blockchair. https://blockchair.com/bitcoin/charts/blockchain-size?compare=ethereum, Nov 2020. [Online; accessed 10. Nov. 2020].

[29] Go Ethereum. https://geth.ethereum.org, Dec 2020. [Online; accessed 3. Jan. 2021].

[30] HiveMQ - Enterprise ready MQTT to move your IoT data. https://www.hivemq.com, Dec 2020. [Online; accessed 3. Jan. 2021].

[31] HiveMQ Extension - File RBAC. https://www.hivemq.com/extension/file-rbac-extension, Dec 2020. [Online; accessed 3. Jan. 2021].

[32] Hyperledger - Open source blockchain for business - IBM Blockchain. https://www.ibm.com/blockchain/hyperledger, Oct 2020. [Online; accessed 13. Nov. 2020].

[33] IBM Food Trust - Blockchain for the world's food supply. https://www.ibm.com/blockchain/solutions/food-trust, Oct 2020. [Online; accessed 13. Nov. 2020].

[34] The future of blockchain - Blockchain Pulse: IBM Blockchain Blog. https://www.ibm.com/blogs/blockchain/2020/04/the-future-of-blockchain, Apr 2020. [Online; accessed 9. Jan. 2021].

[35] Understanding AES 256 Encryption. https://www.solarwindsmsp.com/blog/aes-256-encryption-algorithm, Sep 2020. [Online; accessed 4. Jan. 2021].

[36] What are smart contracts on blockchain? - Blockchain Pulse: IBM Blockchain Blog. https://www.ibm.com/blogs/blockchain/2018/07/what-are-smart-contracts-on-blockchain, Feb 2020. [Online; accessed 17. Dec. 2020].

[37] Workstation Player : Run a Second, Isolated Operating System on a Single PC with VMware Workstation Player. https://www.vmware.com/products/workstation-player.html, Dec 2020. [Online; accessed 3. Jan. 2021].

[38] Cardano. https://cardano.org, Jan 2021. [Online; accessed 14. Jan. 2021].

[39] Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service (S3). https://aws.amazon.com/s3, Feb 2021. [Online; accessed 12. Feb. 2021].

[40] Decentralized Cloud Storage — Storj. https://storj.io, Feb 2021. [Online; accessed 12. Feb. 2021].

[41] Home page | LoRa Alliance®. https://lora-alliance.org, Jan 2021. [Online; accessed 14. Jan. 2021].

[42] Introduction to Smart Contracts — Solidity 0.8.1 documentation. https://docs.soliditylang.org/en/develop/introduction-to-smart-contracts.html, Jan 2021. [Online; accessed 6. Jan. 2021].

[43] Introduction to smart contracts | ethereum.org. https://ethereum.org/en/developers/docs/smart-contracts, Jan 2021. [Online; accessed 5. Jan. 2021].

[44] IoT Growth Demands Rethink of Long-Term Storage Strategies, says IDC. https://www.idc.com/getdoc.jsp?containerId=prAP46737\220, Jan 2021. [Online; accessed 26. Jan. 2021].

[45] Networks | ethereum.org. https://ethereum.org/en/developers/docs/networks, Jan 2021. [Online; accessed 3. Jan. 2021].

[46] SIGFOX.COM. https://www.sigfox.com/en, Jan 2021. [Online; accessed 14. Jan. 2021].

[47] Smart contract languages | ethereum.org. https://ethereum.org/en/developers/docs/smart-contracts/languages, Jan 2021. [Online; accessed 5. Jan. 2021].

[48] The Eth2 upgrades | ethereum.org. https://ethereum.org/en/eth2, Jan 2021. [Online; accessed 14. Jan. 2021].

[49] Muhammad Salek Ali, Koustabh Dolui, and Fabio Antonelli. Iot data privacy via blockchains and ipfs. pages 1–7, 2017.

[50] Muhammad Salek Ali, Massimo Vecchio, Miguel Pincheira, Koustabh Dolui, Fabio Antonelli, and Mubashir Husain Rehmani. Applications of blockchains in the internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 21(2):1676–1717, 2018.

[51] Marti Anderson and Cajo Ter Braak. Permutation tests for multi-factorial analysis of variance. *Journal of statistical computation and simulation*, 73(2):85–113, 2003.

[52] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. *Hyperledger fabric: a distributed operating system for permissioned blockchains*. Association for Computing Machinery, New York, NY, USA, Apr 2018.

[53] Joseph Jose Anthraper and Jaidip Kotak. Security, privacy and forensic concern of mqtt protocol. In *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India*, 2019.

[54] Binance Academy. Sybil Attacks Explained. *Binance Academy*, Nov 2020.

[55] Binance Academy. What Is Staking? *Binance Academy*, Dec 2020.

[56] Francesco Buccafurri, Vincenzo De Angelis, and Roberto Nardone. Securing mqtt by blockchain-based otp authentication. *Sensors*, 20(7):2002, 2020.

[57] Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, 1:22–23, 2013.

[58] Talal Ashraf Butt, Razi Iqbal, Khaled Salah, Moayad Aloqaily, and Yaser Jararweh. Privacy management in social internet of vehicles: review, challenges and blockchain based solutions. *IEEE Access*, 7:79694–79713, 2019.

[59] Yi-Cheng Chen, Yueh-Peng Chou, and Yung-Chen Chou. An image authentication scheme using merkle tree mechanisms. *Future Internet*, 11:149, 07 2019.

[60] Nicholas Confessore. Cambridge Analytica and Facebook: The Scandal and the Fallout So Far. *N.Y. Times*, Nov 2018.

[61] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration. *ACM Comput. Surv.*, 51(6):1–29, Jan 2019.

[62] Ali Dorri, Salil S. Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for IoT security and privacy: The case study of a smart home. *IEEE*, pages 13–17, 2020.

[63] Ali Dorri, Marco Steger, Salil S Kanhere, and Raja Jurdak. Blockchain: A distributed solution to automotive security and privacy. *IEEE Communications Magazine*, 55(12):119–125, 2017.

[64] Filecoin. A decentralized storage network for humanity's most important information | filecoin, 2021.

[65] M. Fischer, D. Kümper, and R. Tönjes. Towards improving the privacy in the mqtt protocol. In *2019 Global IoT Summit (GIoTS)*, pages 1–6, 2019.

[66] Paula Fraga-Lamas and Tiago M Fernández-Caramés. A review on blockchain technologies for an advanced and cyber-resilient automotive industry. *IEEE Access*, 7:17578–17598, 2019.

[67] Saptarshi Gan. An iot simulator in ns3 and a key-based authentication architecture for iot devices using blockchain. *Indian Institute of Technology Kanpur*, 2017.

[68] K. Hantrakul, S. Sitti, and N. Tantitharanukul. Parking lot guidance software based on mqtt protocol. In *2017 International Conference on Digital Arts, Media and Technology (ICDAMT)*, pages 75–78, 2017.

[69] HiveMQ. Enabling the connected car with hivemq. `https://www.hivemq.com/solutions/iot/enabling-the-connected-car/`, October 2020.

[70] HiveMQ. Quality of service 0,1 2 - mqtt essentials: Part 6. `https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/`, October 2020.

[71] Frank T Johnsen, Trude H Bloebaum, Morten Avlesen, Skage Spjelkavik, and Bjørn Vik. Evaluation of transport protocols for web services. In *2013 Military Communications and Information Systems Conference*, pages 1–6. IEEE, 2013.

[72] Michael Kapilkov. Future elections could be held on the Cardano blockchain, says Hoskinson. *Cointelegraph*, Oct 2020.

[73] Ali Safari Khatouni, Martino Trevisan, and Danilo Giordano. Data-driven emulation of mobile access networks. In *2019 15th International Conference on Network and Service Management (CNSM)*, pages 1–6, 2019.

[74] Alexia Dini Kounoudes and Georgia M Kapitsaki. A mapping of iot user-centric privacy preserving approaches to the gdpr. *Internet of Things*, 11:100179, 2020.

[75] Loic Lesavre, Priam Varin, Peter Mell, Michael Davidson, and James Shook. A taxonomic approach to understanding emerging blockchain identity management systems. *arXiv preprint arXiv:1908.00929*, 2019.

[76] Pin Lv, Licheng Wang, Huijun Zhu, Wenbo Deng, and Lize Gu. An iot-oriented privacy-preserving publish/subscribe model over blockchains. *IEEE Access*, 7:41309–41314, 2019.

[77] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[78] R. Neisse, G. Steri, and G. Baldini. Enforcement of security policy rules for the internet of things. In *2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 165–172, 2014.

[79] OASIS. Mqtt version 5.0. `https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html`, October 2020.

[80] Gary W Oehlert. *A first course in design and analysis of experiments*. 2010.

[81] P. Porambage, M. Ylianttila, C. Schmitt, P. Kumar, A. Gurtov, and A. V. Vasilakos. The quest for privacy in the internet of things. *IEEE Cloud Computing*, 3(2):36–45, 2016.

[82] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future generation computer systems*, 88:173–190, 2018.

[83] Markus Schäffer, Monika Di Angelo, and Gernot Salzer. *Performance and Scalability of Private Ethereum Blockchains*, pages 103–118. 08 2019.

[84] Meena Singh, MA Rajan, VL Shivraj, and P Balamuralidhar. Secure mqtt for internet of things (iot). In *2015 Fifth International Conference on Communication Systems and Network Technologies*, pages 746–751. IEEE, 2015.

[85] Dipa Soni and Ashwin Makwana. A survey on mqtt: a protocol of internet of things (iot). In *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*, 2017.

[86] M. Suresh, P. Saravana Kumar, and T. V. P. Sundararajan. Iot based airport parking system. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–5, 2015.

[87] L. M. R. Tarouco, L. M. Bertholdo, L. Z. Granville, L. M. R. Arbiza, F. Carbone, M. Marotta, and J. J. C. de Santanna. Internet of things in healthcare: Interoperatibility and security issues. In *2012 IEEE International Conference on Communications (ICC)*, pages 6121–6125, 2012.

[88] The HiveMQ Team. Creating highly available and ultra-scalable MQTT clusters. `https://www.hivemq.com/blog/clustering-mqtt-introduction-benefits`, Dec 2020. [Online; accessed 3. Jan. 2021].

[89] The Modex Team. A brief history of blockchain, smart contracts and their implementation. *Medium*, Jun 2018.

[90] Pureswaran Veena, Sanjay Panikkar, Sumabala Nair, and Paul Brody. Empowering the edge-practical insights on a decentralized internet of things. *IBM Institute for Business Value*, 17, 2015.

[91] Sandra Wachter. Normative challenges of identification in the internet of things: Privacy, profiling, discrimination, and the gdpr. *Computer law & security review*, 34(3):436–449, 2018.

[92] Electronic Wings. Nodemcu mqtt client with esplorer ide. `https://www.electronicwings.com/nodemcu/nodemcu-mqtt-client-with-esplorer-ide`, October 2020.

[93] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. *arXiv preprint arXiv:1906.11078*, 2019.

[94] A Yakovenko. Solana: A new architecture for a high performance blockchain, 2018.

[95] Mengmeng Yang, Tianqing Zhu, Kaitai Liang, Wanlei Zhou, and Robert H Deng. A blockchain-based location privacy-preserving crowdsensing system. *Future Generation Computer Systems*, 94:408–418, 2019.

Appendices

A Visual validation of certain requirements

A.1 Intermediation between blockchains

In Figure A.1, it has represented the *createLog* event parsed in the private blockchain by the proxy and the consecutive log event created in the public blockchain. In the *Ropsten* explorer, it is possible to confirm the creation of the log event and check the information of its transaction by using the transaction hash, as displayed in Figure A.2. Thus, the interaction between the two blockchains is done successfully through the proxy.

```
Request: createLog      From: 0x8A4de5732c6E8a86dDBb49355443f5163942d71d        To: 0x8A4de5732c6E8a86dDBb49355443f5163942d7
1d      Timestamp: 1970-01-19 18:51:03.285000    Value1: 51a297fa-3613-4781-ba1a-93d6d198b40d   Value2: accident

[Log event created]: 0x763f493c01949dc7929ff8cb1582df96d6c29cedb6bdf4652fb2e44a6ba69948
```

Figure A.1: Request created by the MQTT broker to create a log event in the public blockchain handled by the smart contract proxy.

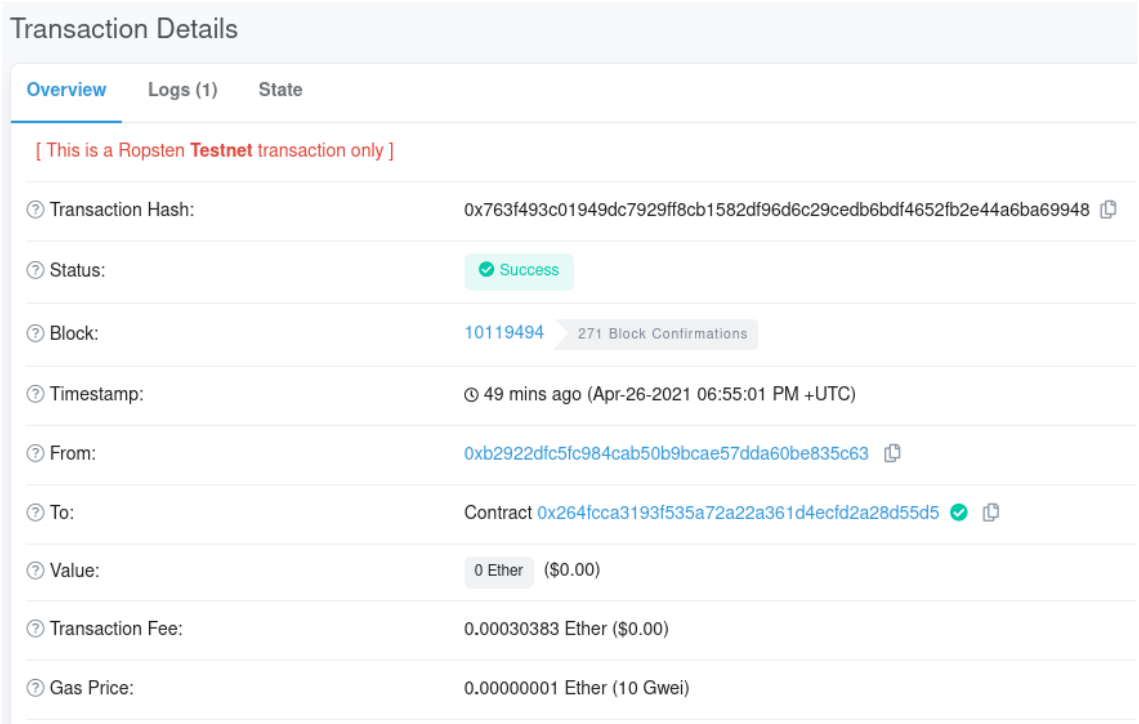

Figure A.2: Information about the log event transaction created in the *Ropsten* tesnet.

A.2 Data management

Since Storj allows deleting and modifying data, data management features were implemented in the system, allowing the user to keep track of the stored data and make modifications or erasures on demand. The proposals in [49, 62, 76, 95] don't offer any data management features. The logs of three possible data management requests (access, modify and delete) are represented in the three flowing figures. It is visible the message sent by the sensor controller to the MQTT broker containing the request and its information in all of them. The succeeding information is the response received, which validate the

successful handling of the requests.

access:

Figure A.3: Request to access all the vehicle users data.

modify:

Figure A.4: Request to modify a specific data entry.

delete:

Figure A.5: Request to delete a specific data entry.

A.3 Access control

The following figure displays a denied data access request made by an agent node of the public blockchain, which validates the access control mechanism in the smart contract of the private blockchain. Figure X displays a successful data access request.

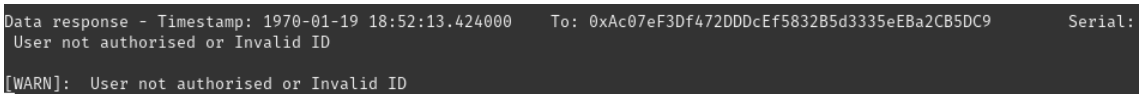

Figure A.6: Denied access request.

A.4 Log information

The subsequent figures contain the logs of a new data log event created in the public blockchain (Figure A.7), the synchronous request made by the agent which was triggered by the log, and the obtained response (Figure A.8).

```
New log - Timestamp: 1970-01-19 18:51:07.948000  ID: 9c51cf17-914e-418f-a288-3f1f94ec3fab        Data Type: accident

[Data request]:  0xe2399550843011f91b3d96d17e0957abd1fd3b6b979b5d247770d42ab24f4cda
```

Figure A.7: New data available log event captured by the agent node, which triggered a data request.

```
Data response - Timestamp: 1970-01-19 18:51:10.807000 To: 0xAc07eF3Df472DDDcEf5832B5d3335eEBa2CB5DC9  Serial: 13DYJsHa
Jbcp7oVuSTKbyNMgBM4QAP4ejcQdf2Ma2iTRyRRrthUVvMgBGmBfomjUbDSiuwZaeDpfjAoVeChwWYNmPK1y7jfPt2ByF8gLAPCvDm6cAEmfKNPDGzPAwe
pbSbppUYTe7U8DHMid3RrNT5eYuqMR5217iknG2P9CpRDjJNdWfGqcXS27EBCaMsx1vWCwksNPSgX7kFtjfDHG8d2EGVXLeHAJNFErumP6aZCqUqdsyade
j4p1zKZokMgGJazrR4nrmpECkWriZ6M98Uk8DoMuMhvVL3hPRNYv1YSbp4UJ4DJiNSCJ69L2fEXs6NFJEqJifMtw6knrr5bXCZT1gxqXaH796qHrdETxQT
Uvj5Fm6jtygA8wpjNBbZExDxBr3GZD13axmqHFzYTopWGeT6YXmU8ExYRS87XzYpuoFgWvDE5iSrkGjhevWbTpqDe2mj6iPjggEU5Pgd2QnCwYjedAHgvw
1qej8o3324QvRvmJQqTN94W3MH2wVaG5V1D7DEk6Sgo1TzEdqqeuFW8SxeLZ6TGvJmDX1AN9oqN5CHrwh7xBX7VZ8eSsUYAHCbRRTrjURTnCvqa9qgB8d7
aRgP2BoX

[WARN]: Serial already acquired for ID  1e79515f-a9ce-47a5-9103-2cef6c01b9a5

[Decrypted Data]:  b'{"timestamp":"2021-04-26T20:58:29.515Z","location":"40.300464, -8.108225","value":"accident"}'
```

Figure A.8: Data response parsed by the agent, and consequent access to the Storj storage and decryption of the data.

A.5 Request handler

Figure A.9 illustrates a data publish request sent by the sensor controller, which was consequently handled. The data content was stored in Storj, and a serial number was generated and stored in the smart contract to be previously used to access the data stored.

```
Topic:  /0x141cEa665Bc9Bd2637f5F5456d611A0Df4B443B2/data/1619472587562  Msg:  {'command': 'publish', 'dataInfo': 'accide
nt', 'value': '{"iv":{"type":"Buffer","data":[209,189,86,237,120,146,59,67,131,243,175,164,168,84,125,10]},"cipher":"864
ba1469768399aec0548924f4ed4f0e5d067905bd5d222d3342e6c1ba45d98dcc2747b3e64d7a7175e71129bd0590fcbc9ebc89bf214752f842911992
dffc78182c72dccffa22824f8be49e561583e18b8fc329cee8b2e38a802c946d18391"}'}       QoS:  1        Retain:  0
Uploaded:  7ed36a5d-fe28-49c7-b485-19a83fab24ca

Generating serialized Access...
```

Figure A.9: Publish data request handled by the MQTT broker.

A.6 Data confidentiality

The following figures display the state of the data in various stages. Figure A.10 displays encrypted data when in transmission between the sensor controller and the MQTT broker. Figure A.11 shows the encrypted data when sent to the MQTT broker in the message, and Figure A.12 displays the data encrypted when stored in the Storj storage system.

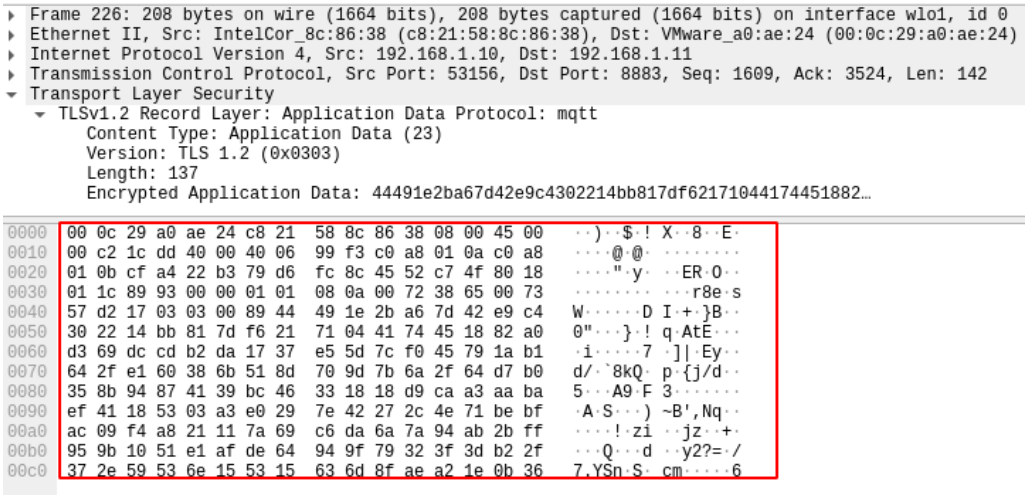

Figure A.10: Data encrypted when in transmission.

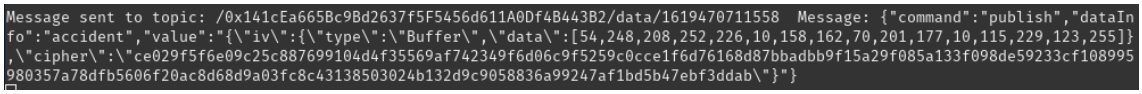

Figure A.11: Data encrypted when received by the MQTT broker.

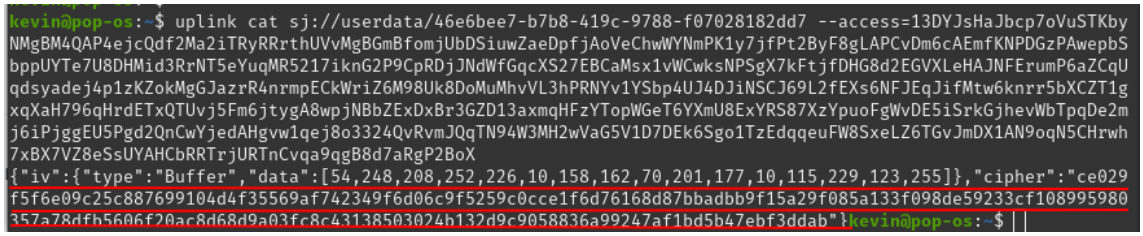

Figure A.12: Data encrypted when stored in Storj.

Authentication The following figure contains a log of an authentication failure in the MQTT broker, where the user is not registered in the RBAC configuration of the broker.

Figure A.13: Authentication failure in the MQTT broker.

Authorisation (Observation) The MQTT client is only authorised to publish and subscribe to certain topics. In this case, the vehicle user tried to publish information on a topic that belonged to another user (Figure A.14).

Figure A.14: Authorisation failure in the MQTT broker.